FINAL REPORT

VOLUME 5

SUMMARY

CLIN 0006

November 2, 1990

MACROSTRUCTURE LOGIC ARRAYS

Contract No. DASG60-85-C-0041

Sponsored By

The United States Army Strategic Defense Command

COMPUTER ENGINEERING RESEARCH LABORATORY

Georgia Institute of Technology

Atlanta, Georgia 30332 - 0540

Contract Data Requirements List Item F006

Period Covered: 1985-1990

Type Report: Final

DISCLAIMER

DISCLAIMER STATEMENT - The views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other official documentation.

DISTRIBUTION CONTROL

(1) DISTRIBUTION STATEMENT - Approved for public release; distribution is unlimited.

(2) This material may be reproduced by or for the U.S. Government pursuant to the copyright license under the clause at DFARS 252.227 - 7013, October 1988.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT 1)Approved for public release; distribution is unlimited. 2) (continued on reverse side)
DECLASSIFICATION/DOWNGRADING SCHEDULE	
PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
NAME OF PERFORMING ORGANIZATION chool of Electrical Eng. eorgia Tech — 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION U.S. Army Strategic Defense Command
ADDRESS (City, State, and ZIP Code) tlanta, Georgia 30332	7b ADDRESS (City, State, and ZIP Code) P.O. Box 1500 Huntsville, AL 35807-3801
NAME OF FUNDING/SPONSORING ORGANIZATION — 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DASG60-85-C-0041
ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO

TITLE (Include Security Classification)

Macrostructure Logic Arrays - Volume 5 - Summary

PERSONAL AUTHOR(S)
C. O. Alford

a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Final	FROM 6/28/85 TO 11/2/90	November 2, 1990	84

SUPPLEMENTARY NOTATION

COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

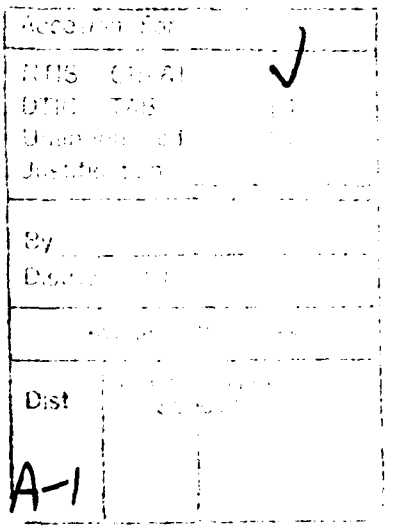
ABSTRACT (Continue on reverse if necessary and identify by block number)

olume 5 - Summary
. Program Overview
1.2 Target/Scene/Seeker Data Generation
1.3 GN&C Processor
1.4 Software
1.5 Digital Emulation Technology Lab
1.6 Parallel Functional Processing
1.7 Payoff
. Parallel Function Processor
2.1 Objectives
2.2 PFP Capabilities
2.3 VLSI PFP
3. Seeker Scene Emulator
3.1 Objectives
3.2 SSE Capabilities
3.3 Advanced SSE
4. Guidance Navigation & Control Processor
4.1 Objectives
4.2 Capabilities
5. Software Development
5.1 Objectives
5.2 Capabilities
5.3 Current Status of the Software Development (cont.

DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	Unclassified
a NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (Include Area Code) — 22c OFFICE SYMBOL

Form 1473, JUN 86 — Previous editions are obsolete — SECURITY CLASSIFICATION OF THIS PAGE

Unclassified



FINAL REPORT
VOLUME 5
SUMMARY

CLIN 0006

November 7, 1990

Author
Cecil O. Alford

COMPUTER ENGINEERING RESEARCH LABORATORY
Georgia Institute of Technology
Atlanta, Georgia 30332 - 0540

Eugene L. Sanders	Cecil O. Alford
USASDC	Georgia Tech
Contract Monitor	Project Director

TABLE OF CONTENTS

1. PROGRAM OVERVIEW

USASDC Contract DASG60-85-C-0041 was initiated June 28, 1985. The objectives of the contract were to develop parallel processing technology to support the USASDC EXO program. This eventually led to four specific areas of research and development.

1.1. Real Time Simulation and Testing

The design of new interceptors requires thousands of simulation runs. The Georgia Tech parallel computer, SPOCK, was explicitly designed to execute such simulations in real time. Further, the SPOCK computer was designed to support external hardware for real time testing and verification. SPOCK became the starting point for a development effort which led to the Parallel Function Processor (PFP).

1.2. Target/Scene/Seeker Data Generation

In order to run a simulation on the PFP it is necessary to have data which represents real targets in a real background. Further the sensor (IR Focal Plane Array) must be modeled to generate the data in a realistic manner. This led to the development of the Seeker/Scene Emulator (SSE) as a real time data input device.

1.3. GN&C Processor

Since the processing requirements for KEW EXO interceptors were becoming much more demanding, a task was initiated to design a GN&C Processor using the same parallel computing methods employed in the PFP. An eight processor system was defined, new VLSI chips were designed and fabricated, leading to a GN&C Processor with exceptional performance.

1.4. Software

Several simulations have been run on the PFP. Currently a 6 DOF EXO Simulation (EXOSIM) is being translated from a serial Fortran program to parallel Ada. Portions of this simulation are now running on the PFP.

An extensive Ada development program is underway to produce tools and compilers to support all the hardware with Ada source code. Compilers for the PFP are already working. Compilers for the GN&C processor will be completed under another contract.

1.5. Digital Emulation Technology Laboratory (DETL)

The four program elements are combined in Figure 1.1 to make up the DETL. All of this work was started under the contract and is being continued under USASDC Contract DASG60-89-C-0142. The following section discusses the parallel functional processing concept which is used as a basis for system design and programming.

1.6. Parallel Functional Processing

The Macrostructure Logic Arrays program is based on the parallel functional processing concept. This concept implies the direct translation of block diagrams to parallel computing structures. The concept can be applied at all levels of KEW processing. These include simulation of complete systems and implementation of guidance and control functions into a G&C processor.

An application example is shown in Figure 1.2. The block diagram which represents a system model is composed to ten computational blocks. The parallel processing implementation consists of ten processing units and a crossbar switch which interconnects the processors. As shown in the figure, each physical block is mapped directly onto a processing unit. The processor in then loaded with a program which represents the mathematical model of the physical block. After all the processing units have been loaded, a connection pattern is loaded into the sequencer which controls the crossbar switch. The switching pattern is a direct mapping of the block connections, which exist in the block diagram, onto the crossbar switch sequencer in the parallel processor implementation. For example, block 3 has outputs to blocks 4 and 5 and inputs from blocks 2 and 10. When this information is compiled for all the blocks, the sequencer can set up the necessary transfers between each processor.

After all program loading is completed, the sequencer assumes control of processing and initiates program execution. Each processor solves its equations for a fixed time increment. On completion, the processor signals the sequencer that data is available for transfer. When the receiving processor is ready for data, the sequencer invokes a transfer. After all transfers are complete, a new compute cycle begins. Since the switching network is a full crossbar, several transfers can occur in parallel. This usually limits the required number of transfer cycles to three or four on each processing step.

The processing concept illustrated in Figure 1.2 is similar when applied to the PFP or to the GN&C Processor. The three basic ideas of multiple (not necessarily identical) processing units, a crossbar switch for interconnection and a sequencer for control are used in each implementation.

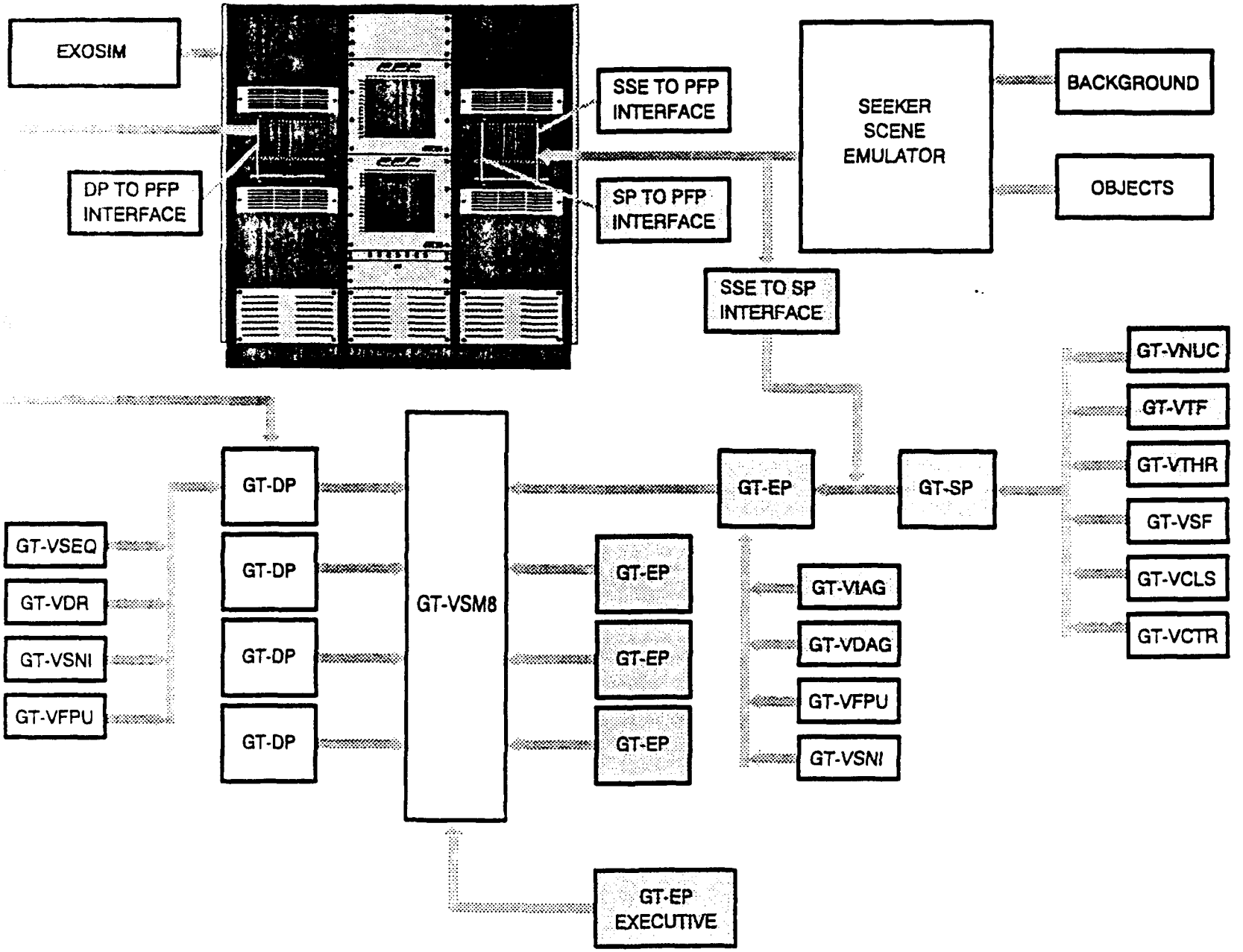


FIGURE 1.1 DIGITAL EMULATION TECHNOLOGY LABORATORY

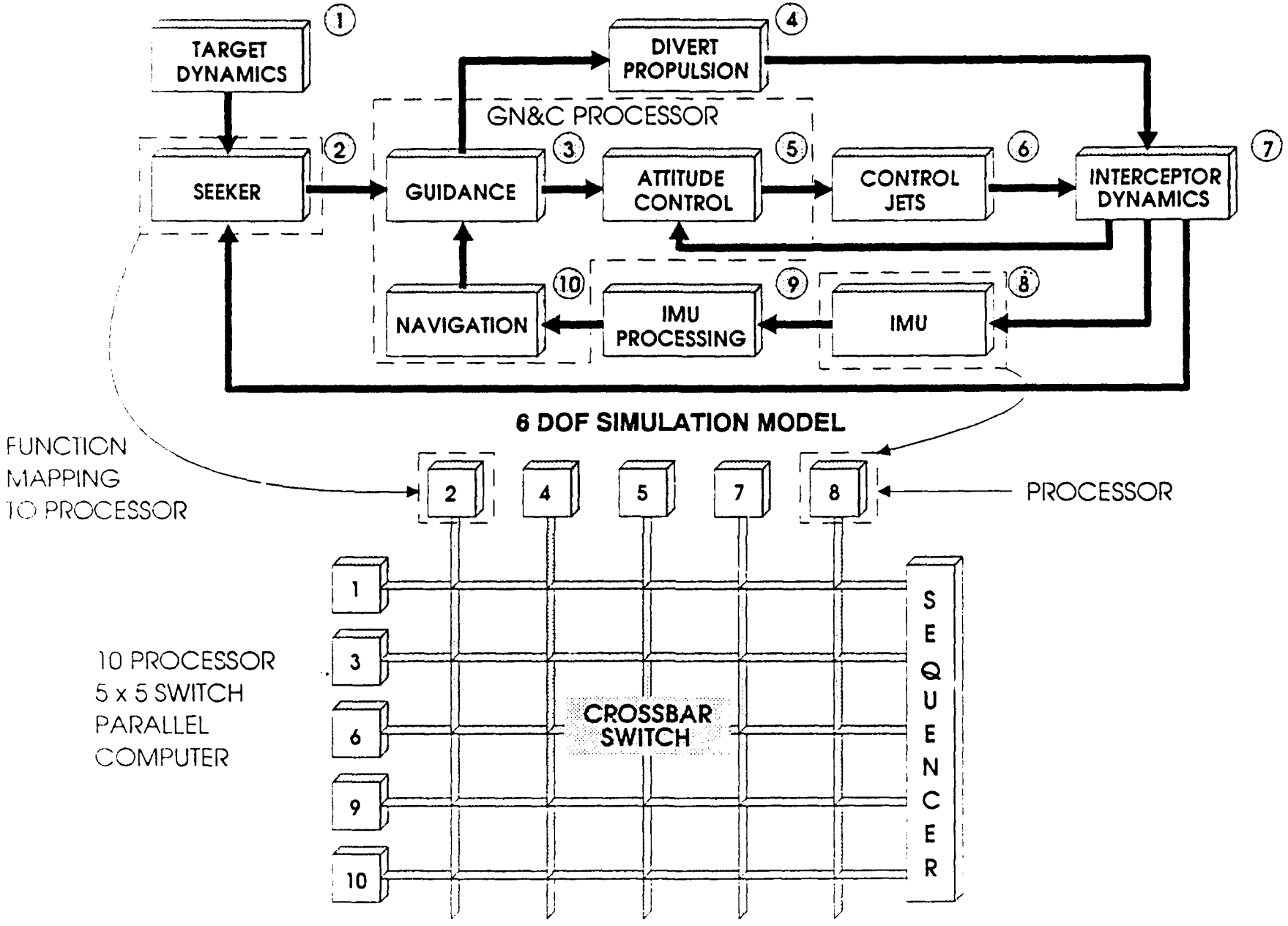


FIGURE 1.2 PARALLEL FUNCTIONAL PROCESSING

Changes are made in the number of processors, crossbar switch structure (serial/parallel) and processor type (fixed point/floating point/special function). The basic structure, as well as the differences, are discussed in the following sections.

1.7. Payoff

There are three payoffs for the DETL technology. These payoffs are in more exact simulations, lower testing costs and spin-offs to non-SDI applications.

Simulation and Testing

The PFP and SSE provide a unique capability to test KEW models, components, and algorithms, at a level of complexity much greater than existing computer systems. Excluding the gamma event problem, threats can be generated which contain all the features a KEW interceptor would expect to see. The ability to generate noise characteristics, large numbers of differing objects, and nonlinear characteristics for each detector pixel, for large FPAs at reasonably high frame rates is astounding. Canned scenes can be generated, stored and used as benchmarks for testing algorithms or flight processors. None of the testing requires cold chambers or any special hardware. Only one interface has to be built; a connection from the SSE output to the processor input.

When the SSE is coupled to the PFP, closed loop simulation is possible for a complete end-to-end KEW mission. The KEW model can be rapidly changed to test the effects of various "what if" conditions. Real components can be inserted into the simulation to replace simulated hardware and software. All of this can be done in a design environment, fully under the control of the designer/user. There is essentially no penalty for additional testing. Hence, more testing and experimentation can be done to arrive at better solutions and to avoid costly design errors.

Silicon Compiler Design

The GN&C Processor has been designed using the Genesil silicon compiler. This is the first KEW processor to take advantage of this cost savings approach to VLSI design. The designs can be used to follow progression of technology from 1.25 macron to 0.8 to 0.5 micron feature sizes. The designs can also be used to move from vendor to vendor, including newer rad-hard processes. These compiled designs will yield large cost benefits in future hardware changes and implementations.

Spin-Offs

It is readily apparent that simulation and testing of KEW models and components is not radically different from many other applications. It is also obvious that other systems use staring arrays to acquire

information which must be processed in a manner not too different from a KEW interceptor. Because the technology is broad based, it can be used in many other military systems and some non-military systems. The key words to look for in an application match are scene generation, real-time simulation, staring arrays, and signal/object/data processing. Systems which use one or more of these key words, are candidates for this technology.

2. PARALLEL FUNCTION PROCESSOR

2.1. Objectives

A critical KEW/SDIO issue is real-time testing of interceptor hardware and software. Testing has to support Focal Plane Array Seekers, GN&C Processors and flight software. The test facility must have capability for parametric studies, large volume data analysis and simple hardware interfaces. System models must be processed in real time with no sacrifice in fidelity. All of the software must be Ada Compatible.

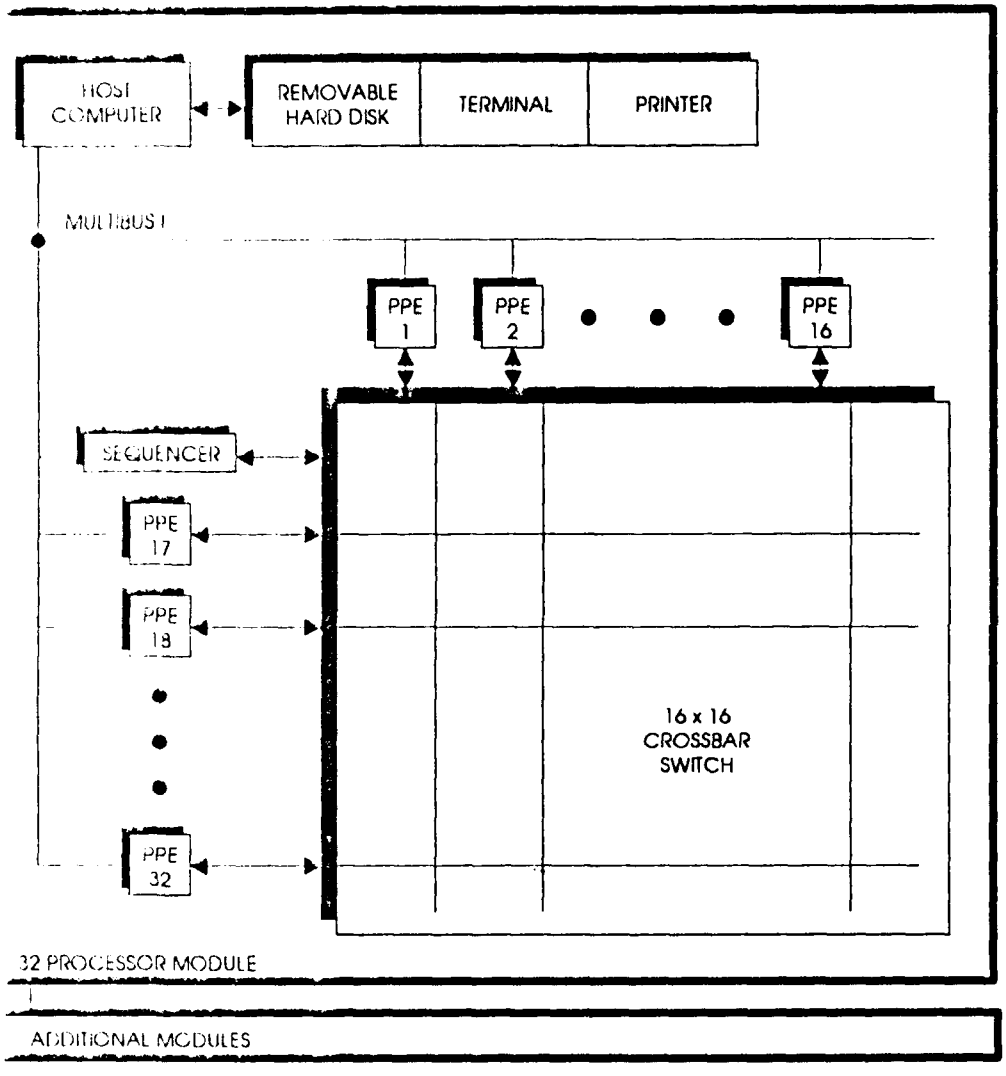
Current real-time simulation hardware fails to meet these requirements. The best computers will execute the required models at less than 100 hertz. In order to achieve accurate real-time results a 2000 hertz bandwidth is necessary. This bandwidth has to be achieved while executing nonlinear and transcendental functions and providing a minimum of eight high speed data channels for hardware under test. The number of ordinary differential equations needed for a KEW 6 DOF model is 60 to 120. All of these requirements are met in the PFP.

2.2. PFP Capabilities

The Parallel Function Processor was designed to meet all requirements of real-time simulation and testing. The computer features 64 processors connected by two crossbar switches. Special purpose computing modules have been designed for hardware interfacing, nonlinear functions and transcendental functions. Interfaces to the SSE and GN&C Processor are in progress. PFP software is being developed to run on a SUN 386i host computer under the UNIX operating system. Since each processor has a throughput of 8 MFLOPS, the system throughput is greater than 480 MFLOPS. Further, additional units can be interconnected to double or triple the number of processors and the resulting throughput.

2.2.1. PFP Architecture

As shown in Figure 2.1, one PFP unit contains 32 processors, a crossbar switch, a host computer and peripherals to support the host. Mapping of functions from the simulation block diagram to the processors is identical to the example in Section1.6. One-to-one functional mapping is a distinct advantage in debugging hardware or software problems. If an engineer wishes to change a parameter in the Boost Autopilot, it is a simple matter to select the appropriate processor which contains the Boost Autopilot model, examine the code, change the parameter, compile the new code and execute the new simulation. Blocks can be added or deleted by adding or deleting processors. The block connections are mapped into the crossbar switch connection code. To delete a block only requires changes in the



UNIQUE PFP FEATURES:

- SUPPORTS 1-TO-1 MAPPING OF INTERCEPTOR FUNCTIONS TO PROCESSORS FOR EMULATION
- CROSSBAR PROVIDES 1-TO-1 IMPLEMENTATION OF FUNCTIONAL COMMUNICATIONS
- EASE OF IMPLEMENTATION AND VERIFICATION OF FUNCTIONAL CHANGES
  - FUNCTION ASSIGNMENT TO PROCESSOR
  - COMMUNICATIONS ASSIGNMENT TO CROSSBAR
- PROCESSING ELEMENTS ARE ARCHITECTURALLY INTERCHANGEABLE AND EXPANDABLE
  - A/Ds, SP, DP, I/O INTERFACES, ETC.
- EASE OF INTERFACING TO OTHER REAL-TIME SYSTEMS

FIGURE 2.1 PFP ARCHITECTURE

MG - \WIN30\GORDON\FR1990/PFPARCH.DRW

connections for those processors which sent data to or received data from the designated processor (block). The crossbar code is then modified, compiled, loaded and ready to run.

The unit shown in Figure 2.1 only represents one-half of a PFP. Another unit contains 32 processors and a crossbar switch. The same host is used for both units. These two units can be run separately, solving two independent problems, or they can be interconnected to solve one very large problems. Interconnection only requires the replacement of one processor on each unit with an Array Interconnect Board. The Array Interconnect Boards are tied together with a data cable which transfers data between any of the processors in one unit to any of the processors in the other unit. More than one Array Interconnect Board set can be used to increase the data rate between the two units.

A typical PFP Hardware Description is given in Figure 2.2. New items which appear are piggyback function boards and analog I/O boards. The function boards are special purpose auxiliary boards to increase processing speed when executing transcendental functions, and table look-ups. Most simulations have a large number of transcendental functions, 10 or more single variable table look-ups, and 6 or more two variable table look-ups. Analog I/O boards are useful to connect external hardware which requires analog input signals and delivers analog output signals. Each Analog I/O board provides 4 input and 4 output channels.

2.2.2. Performance

Several processing elements have been constructed for the PFP. Each of these has been benchmarked solving a second order differential equation using a fourth order Runge-Kutta method. The step size is fixed at 50 steps for the highest frequency that can be solved in real-time. If f_{max} is the bandwidth in hertz and TSTEP is the real-time integration step size in seconds, then

$$f_{max} = 1/(50 * TSTEP). \tag{1.1}$$

Results for this benchmark are shown in Figure 2.3. Five PFP processing elements are benchmarked for comparison. The iSBC86/12 is an old processor typical of IBM PC computers. The iSBC 286/12 is typical of PC/AT type computers and the iSBC 386/12 compares to newer PC type machines. The INMOS T800 is a single chip "Transputer" which is on a board containing four processors. This is a very high performance processing element and is used in other program tasks. The Georgia Tech designed GT-FPP/3, floating point processor, is the same size as the iSBC boards, but uses different technology and a different architecture. This board serves as a basis for the GN&C architecture. One shortcoming of this board is a rather small memory. Additional memory modules have been designed as piggyback boards for this processor. This benchmark does not use any nonlinear or transcendental functions which severely reduces the bandwidth. However, the GT-FPP/3 has an auxiliary function processor board to assist in these calculations, and is much faster than the other processors for

HARDWARE DESCRIPTION

- **THREE-RACK ASSEMBLY**
  - TWO CROSSBAR INTERCONNECTION ARRAYS
  - TWO CROSSBAR SEQUENCERS
  - POWER SUPPLIES AND CARD CAGES
- **HOST COMPUTER**
  - TERMINAL
  - PRINTER
  - REMOVABLE HARD DISK DRIVE
- **64 FLOATING POINT ROCESSORS**
  - 64 PIGGYBACK MEMORY BOARDS
- **TWO TYPES OF PIGGYBACK FUNCTION BOARDS**
  - TRANSCENDENTAL FUNCTIONS: 55 BOARDS PROVIDED
  - LOOK-UP FUNCTIONS: 10 BOARDS PROVIDED
- **5 ANALOG-TO-DIGITAL I/O BOARDS EACH HAVING**
  - 4 CHANNELS FOR INPUTS
  - 4 CHANNELS FOR OUTPUTS
- **10 ARRAY INTERCONNECT BOARDS AND CABLES**
- **SPARE BOARDS**
  - CROSSBAR (1)
  - MEMORY (4)
  - A/D -D/A (1)
  - PROCESSOR (4)
  - FUCTION (4)
  - ARRAY INTERCONNECT (2)

FIGURE 2.2 TYPICAL PFP HARDWARE DESCRIPTION

MG - \WIN30\GORDON\FR1990/PFPHDESC DRW

PFP PROCESSING ELEMENTS					
BOARD	**PROCESSOR**	**CLOCK SPEED (MHz)**	**CO-PROCESSOR**	**MEMORY (KBytes)**	**ODE BANDWIDTH (Hz)**
iSBC86/12	INTEL 8086	5	8087	32	10.2
iSBC286/12	INTEL 80286	8	80287	1000	16.3
iSBC386/12	INTLEL 80386	16	80387	1000	70
B003 (1/4)	INMOS T800	20	N/A	256	643
GT-FPP/3	AMD 29325	8	N/A	176	2520

- ODE USED FOR COMPARISON IS A HIGH-FIDELITY SINE WAVE GENERATOR
- RK-4 INTEGRATION METHOD WITH 50 STEPS PER HIGHEST FREQUENCY
- B003 HAS FOUR PROCESSORS PER BOARD
  - ONLY ONE IS USED FOR TEST

FIGURE 2.3 RELATIVE PERFORMANCE OF PROCESSING ELEMENTS

MG - \WIN30\GORDON\FR1990/RELPERFM.DRW

problems of this type. The benchmark clearly demonstrates the ability of the GT-FPP/3 to meet the computing requirements.Another benchmark, comparing the PFP to well known supercomputers is shown in Figure 2.4. This benchmark is based on published computing times for individual instructions for each computer type. The benchmark problem is again the solution of a set of linear ordinary differential equations. In this case a fourth order Runge-Kutta method is used with a step size of 20 microseconds. This represents 50 steps for at a bandwidth of 1000 hertz. The computation speed of each computer was evaluated to determine how many equations could be solved with this step size in real-time. A typical 3-DOF simulation requires around 50 such equations while a 6-DOF model will require more than 120. The results in Figure 2.4 are for linear equations. No penalty factor has been inserted for transcendental or nonlinear functions. Note the supercomputers are able to use vector chaining for linear problems but are reduced to scalar performance as nonlinear elements are added. Thus, a realistic comparison, solving a real 6-DOF model would show a reduced performance for the supercomputers but little degradation for the PFP.

2.3. VLSI PFP

A newer design for the PFP is underway. This design is based on the current PFP, but uses VLSI designs to implement the crossbar in a more compact manner. Sixty-four processors will be connected using one Fully Connected Switch. The host computer will be a SUN running the UNIX operating system. The processing elements will be based on the Intel i860 processor. Expected performance for this computer will be a sustained 500 MFLOPS with a peak rate exceeding 2500 MFLOPS. Cabinet size, as shown in Figure 2.5, is 48" x 22" x 26". This system design is continuing under USASDC Contract DASG60-89-C-0142.

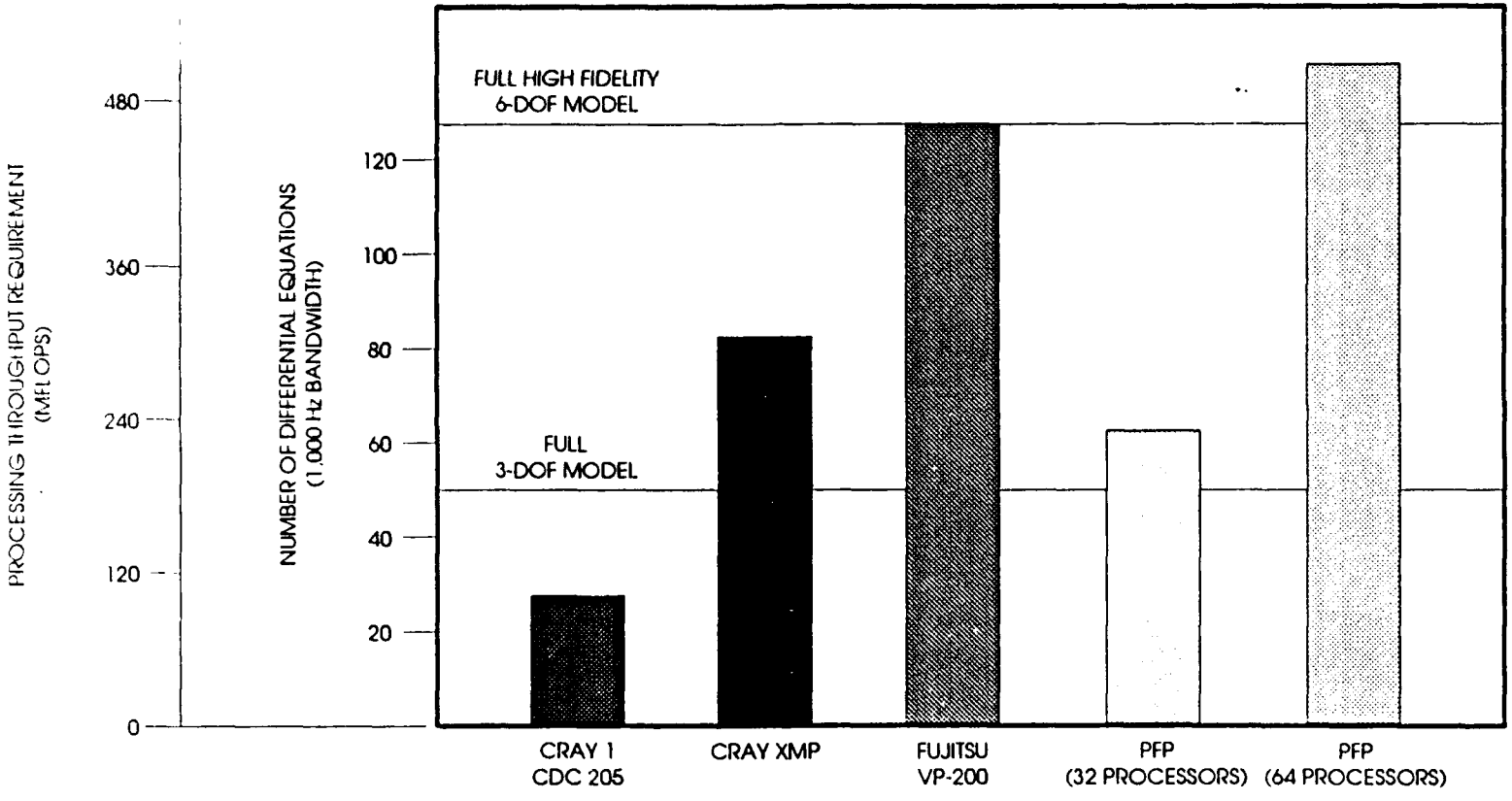


FIGURE 2.4 RELATIVE PFP PERFOMANCE

MG - \WIN30\GORDON\FR1990/PERFORM.DRW

- 64 PROCESSOR SYSTEM CABINET WITH SUN HOST
- FULLY CONNECTED 64 PROCESSOR CROSSBAR FOR HIGHER FUNCTIONAL THROUGHPUT
- 53.00" x 22.25" x 25.50"
- PROVIDE IN EXCESS OF 500 MFLOP PROCESSING CAPABILITY

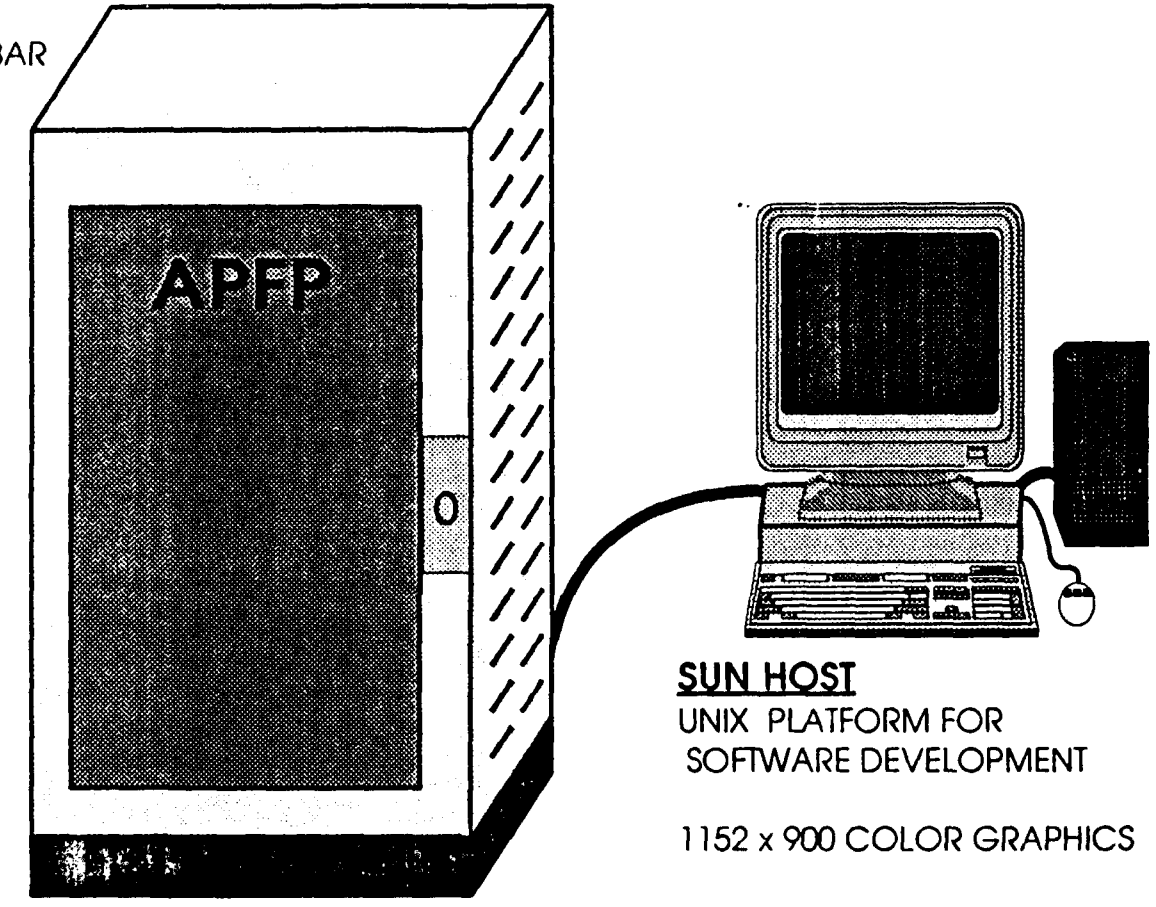


FIGURE 2.5 APFP VLSI PROCESSOR

MG - \WIN30\GORDON\FR1990/VLSIPROC.DRW

3. SEEKER/SCENE EMULATOR

3.1. Objectives

The SSE is to be used as an input device to 6-DOF simulations. Since it will model the staring array seeker it must be capable of generating real-time data, at the pixel level. The driving requirements for data are traceability to the Phase One Threat Scene (POTS) which requires the use of OSC threat tapes. in addition, the background data must include the effects of Operation In a Nuclear Environment (OPINE). This requires accurate modeling of redout and debris gamma.

The seeker to be modeled is a 128 x 128 staring array with non-uniformity characteristics on each pixel. Frame rates are to vary from 0 to 100 frames/second. Pixel data is to reflect the effects of A/D conversion. Signal processing algorithms such as non-uniformity correction and thresholding are to be available for use when required.

The architecture must support more than one seeker model. While it is to be used in connection with the LATS and LETS programs, these are not the only end uses. Hence any staring array seeker model should be a candidate for use as a data generation model.

Threat models must cover all types; targets, decoys, and objects. Further, the number of objects in the Field-of-View (FOV) should be virtually unlimited. If real testing is to be done on GN&C processors it must be possible to generate realistic data for large numbers of objects at real-time rates.

3.2. SSE Capabilities

The Seeker/Scene Emulator has been designed to meet all the objectives stated in Section 3.1. Some of the nuclear modeling needs to be improved, multi-spectral capability needs to be added and more noise sources should be used. These features are reserved for a later upgrade.

The current SSE can deliver in real-time the features shown in Figure 3.1. The most important is a scalable design which can be extended to give higher frame rates and more frames of data.

Data is modeled over a 128 x 128 grid. Electrons are generated as a function of intensity for several effects. The models present in the SSE data are (1) fixed-pattern noise, (2) nuclear effects, (3) FPA/seeker and (4) objects. Each of these can be modeled with a wide variety of parameters. The list for each is given in Figures 3.2 - 3.5. This is an enormous capability when coupled with the ability to do individual pixel data construction at real-time rates of 128 frames/second.

- Variable Frame Rate up to 128 Frames/sec
- Variable Timeline up to 480 Frames of Data
- Any Number of Objects
- Programmable Real-Time, 5-Point Pixel Response Capability per Pixel
- Programmable Real-time A/D Emulation (Bits, Scale, Reference)
- Architecture Scalable to Higher Performance Levels

Figure 3.1 Current Emulator Capabilities

Noise Sources Currently Include:

- Blackbody Emissions
  - Window
  - Mirror(s)
  - Baffle
  - Filter
- Empty Space Background Radiation
- Ambient Gamma Radiation

Figure 3.2 Fixed-Pattern Noise Parameters

- Gamma Shielding Coefficient
- Gamma Circumvention Coefficient
- Gamma Event Cross Section
- Gamma Event Charge Multiplier
- Number Of Radiation Scenarios
- Gamma Fluxes For Each Scenario
- Redout Emittances For Each Scenario
- Background Emittances For Each Scenario
- Radiation Scenario Number

Figure 3.3 Nuclear Effects Parameters

- Aperture Diameter
- F-Number
- Obscuration Ratio
- Transitivity

- Window Temperature
- Window Radius
- Window Distance

- Number Of Mirrors
- Temperature Of Each Mirror
- Emissivity Of Each Mirror

- Number Of Pixels In Both Dimensions
- Number Of Image Subpixels For Each FPA Pixel
- Pixel Spacing In Both Dimensions
- Fill Factor In Both Dimensions
- Quantum Efficiency, Average
- Quantum Efficiency, Std. Deviation
- Offset, Average
- Offset, Std. Deviation
- FPA Temperature
- D Star
- Band Gap Of FPA
- Optics Aberrations

Figure 3.4 FPA/Seeker Model Parameters

- Unlimited number of objects
- Objects move with respect to each other
- Objects can grow or diminish
- Arbitrary rotation
- Object eclipsing

Object intensity computation:

- Assumes blackbody radiation
- User-specified
  - Temperatures
  - Emissivities
  - Waveband limits
- Currently supported object types:
  - Simple conic object
  - Conic object with differently radiating base
  - Simple spherical object
  - Spherical object with "stripes"

Figure 3.5 Object Model Parameters

3.2.1. SSE Architecture

All data is generated off-line. A simulation trajectory is run from end-to-end in a non-real-time mode. The seeker line-of-sight (LOS) and field of view (FOV) are used to specify the background area and objects. The background data is separated from the object data for each frame. In the real-time simulation, the PFP runs the same simulation and the same trajectory. The line-of-sight may vary slightly from the original non-real-time run. To compensate for this shift in LOS the background data is fixed to the new LOS but the target data is moved across the background data to reflect the change in LOS from the initial run to the real-time run.

In Figure 3.6 each object is mapped onto a 128 x 128 array using the LOS from the 6-DOF simulation and optical parameters from the seeker. Each object pixel is expanded into a 4 x 4 array to develop high resolution data for the object. Object intensity at the dector pixel is used to generate electrons by combining the factors listed in Figures 3.2 - 3.5. The object data is mapped onto a frame image. This data is converted to the frequency domain using a 2D-FFT. The converted image data is multiplied by a Modulation Transfer Function to account for seeker optical parameters. The resulting image, after an inverse transform, is a single wavelength, 128 x 128 object image, which is fully diffracted and aberrated.

In operation the sequence of events proceeds as shown in Figure 3.7. Data is generated off-line using (1) a seeker model, (2) an engagement scenario, and (3) a 6-DOF simulation. The background and object data are compiled for each frame of flight. When the data is loaded into the SSE and played into the signal/data processor and the PFP running the 6-DOF simulation, a new line of sight is generated. This data is sent to the SSE to correct for the original LOS by moving the objects over the background.

The computational requirements are illustrated in Figure 3.8. This figure ignores all shifting operations and merely gives floating point operations to accomplish pixel output calculations. The output is a fixed point 16 bit value which emulates most A/D converters for staring arrays.

The SSE design parameters are shown in Figure 3.9. The major items are the large storage and the very large data rates required for the objects. This is accentuated by the sub-pixel resolution used to enhance target fidelity.

The architecture shown in Figure 3.10 has been designed, built and tested to meet all requirements of Figure 3.9. The objects are stored in a distributed memory on the left of the figure in the section designated object formatter. Frames of data are rolled into a bank of Transputer processors. LOS information is used to shift the frame data in two dimensions.

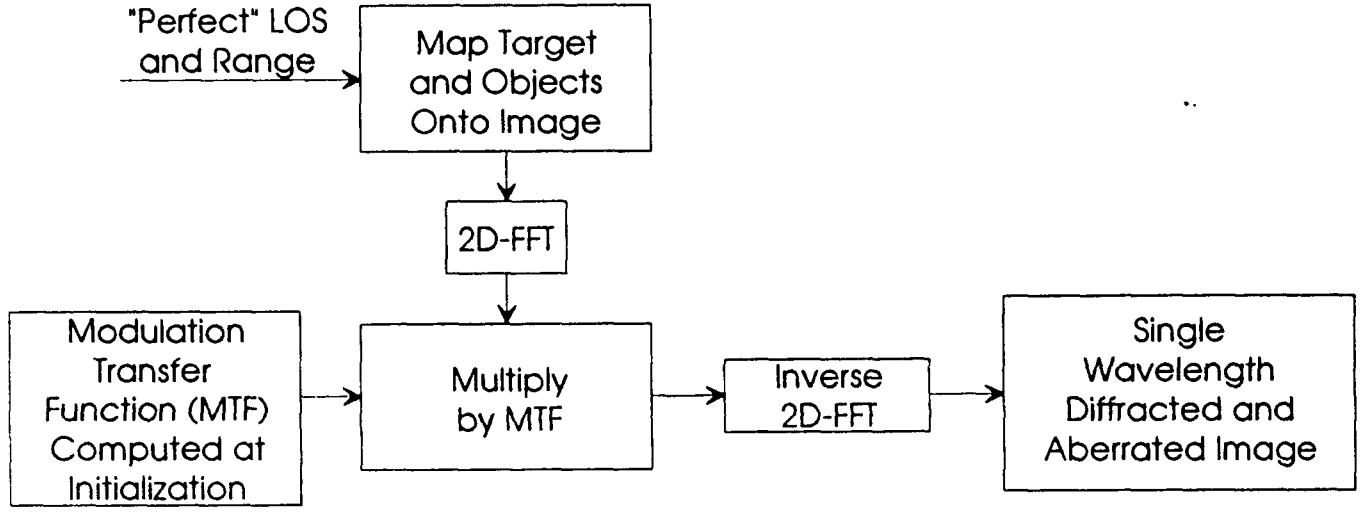


FIGURE 3.6 TARGET/OBJECT DATA CONSTRUCTION

MG - \WIN30\GORDON\FR1990/T-O_DATA.DRW

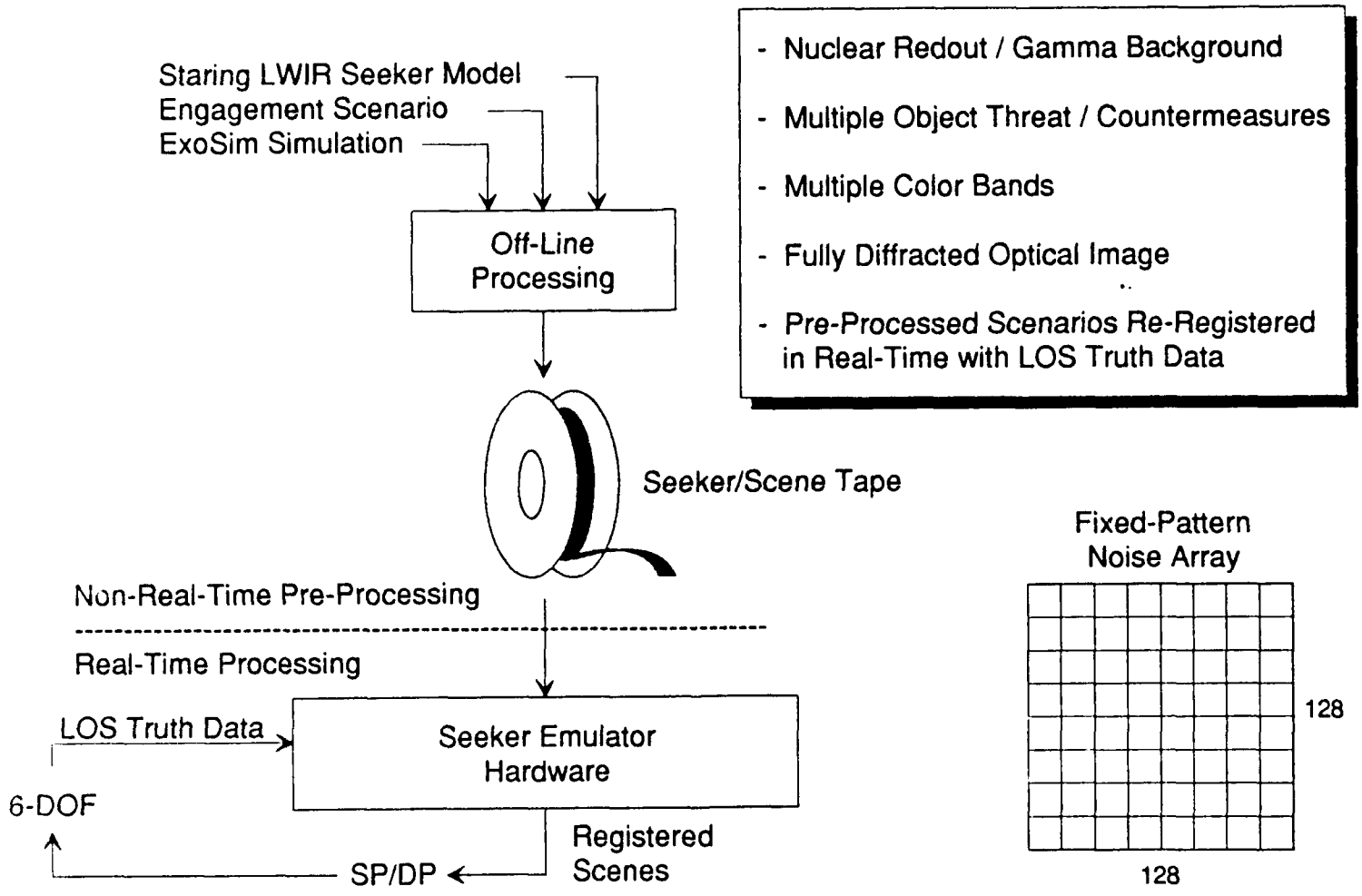


Figure 3.7 Data Generation and Reconstruction

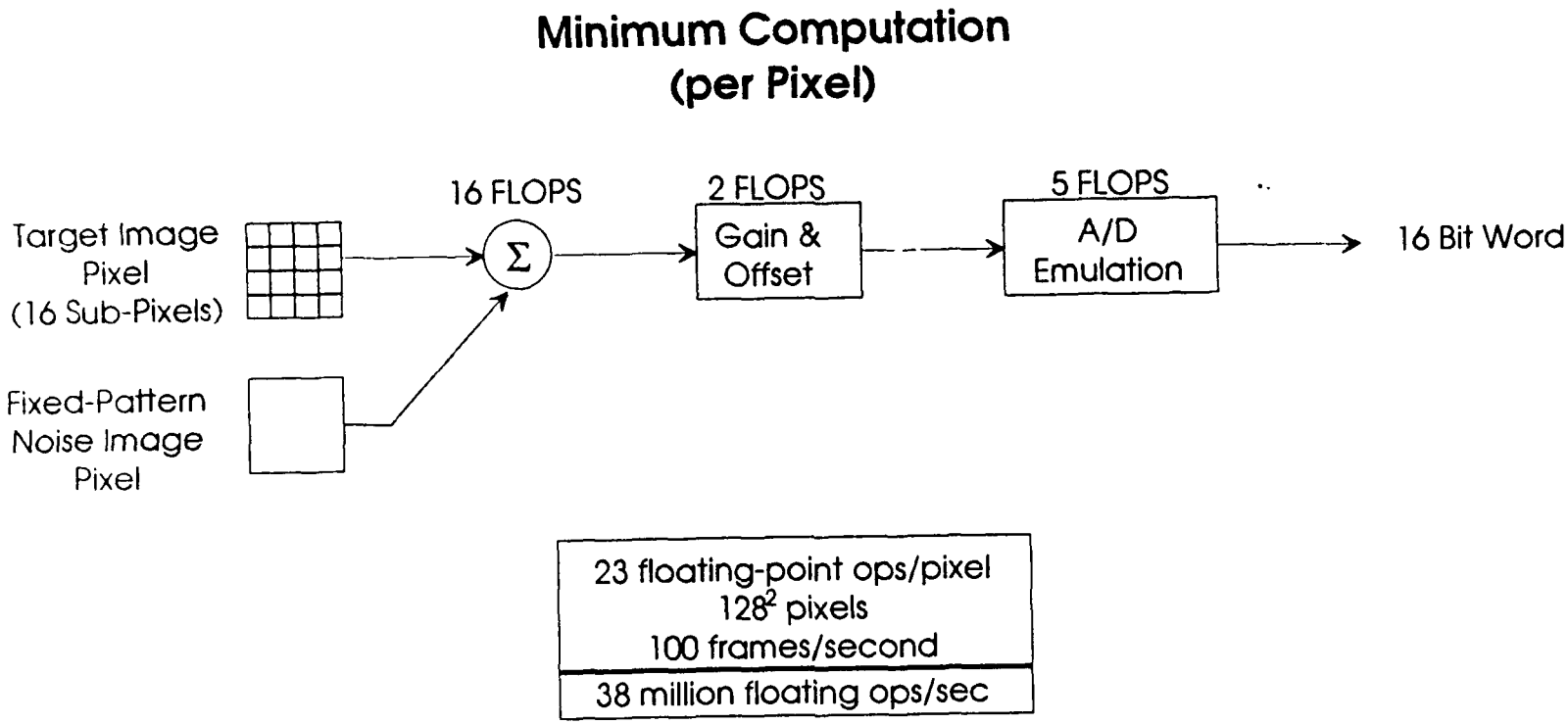


FIGURE 3.8 SSE COMPUTATIONAL REQUIREMENTS

MG - \WIN30\GORDON\FR1990/SSE-COMP DRW

SEEKER REQUIREMENTS

Rates	up to 100 frames/sec
FPA size	128 X 128
Total frames	~ 400
Word Size	up to16 bits

BACKGROUND IMPLEMENTATION REQUIREMENTS

Storage requirements	~ 13 megabytes
Data throughput	3 MB/SEC

OBJECT/TARGET IMPLEMENTATION REQUIREMENTS

Sub-pixel resolution	4 x 4
Storage requirements	~ 200 megabytes
Data throughput	52 MB/SEC

Figure 3.9 SSE Design Parameters

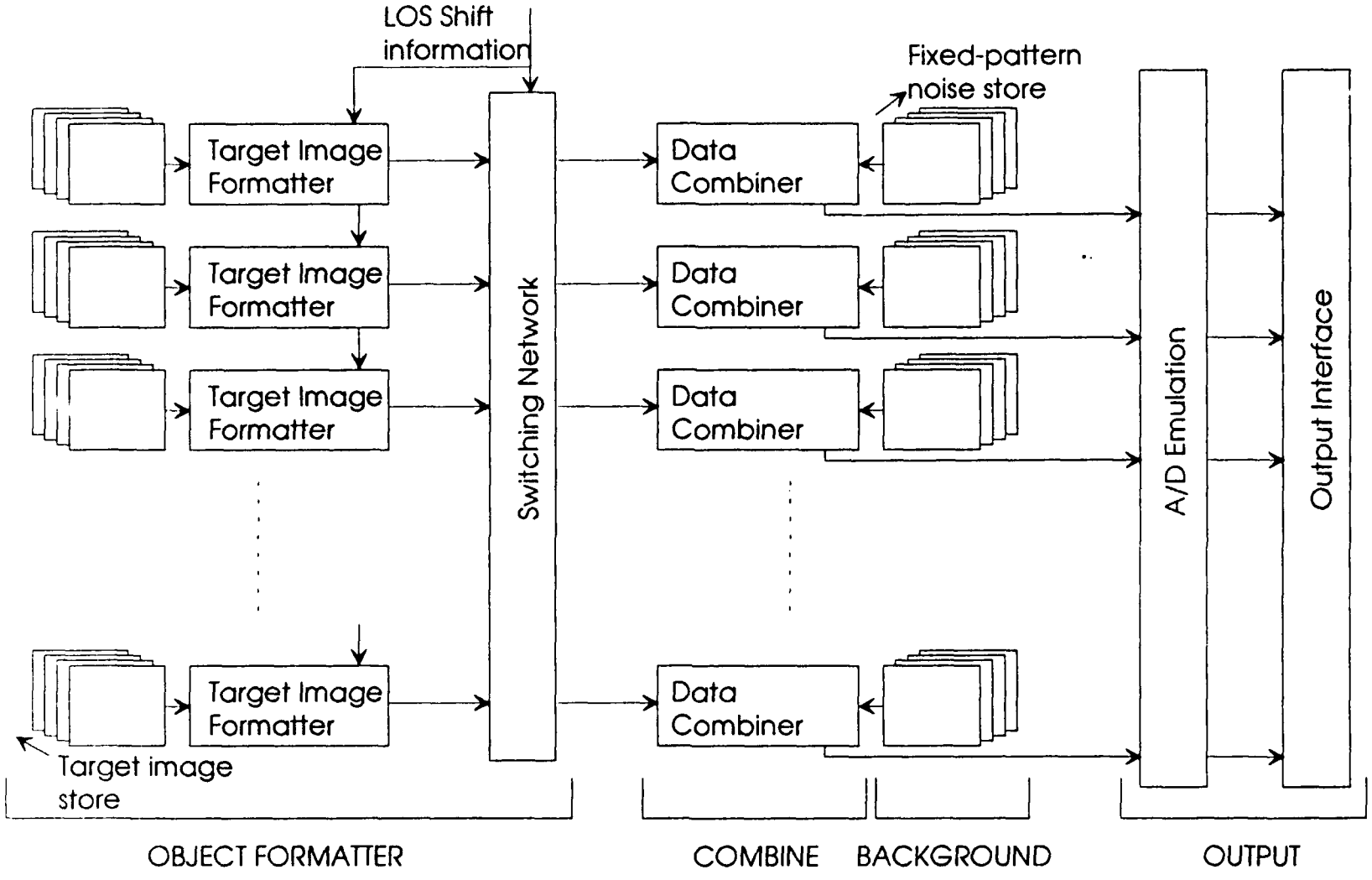


FIGURE 3.10 SSE ARCHITECTURE

MG - \WIN30\GORDON\FR1990/SSE-ARCH.DRW

While this operation takes place, background data is rolled out of a separate distributed memory into the data combiner. The object data and background data are combined and then output to the A/D model. After "A/D conversion", 16-bit fixed point data is output for each pixel in the frame.

3.2.2. SSE Performance

The SSE system is shown in Figure 3.11. In addition to the object generator and background generator there is a (1) Microvax II, (2) PC-AT based host, (3) image processing, (4) image display, and (5) and interface to the PFP. These peripherals are detailed in Figure 3.12. All programming has been done in OCCAM, but compilers exist for Fortran, Pascal and Ada. Georgia Tech is currently running tests on the Ada compiler.

These components have been built and tested, operating in an open-loop and closed-loop configuration. The open-loop configuration shown in Figure 3.13, demonstrated 128 x 128 emulation capability at 128 frames per second. The display cannot be updated at this rate, and was refreshed at a reduced rate. The SSE has been connected to the PFP in a closed-loop demonstration. The only remaining interface which has not been demonstrated is the connection to a Signal Processor. This is on-going under USASDC Contract No. DASG60-89-C-0142

3.3. Advanced SSE

Two major problems exist with the current implementation of the SSE system. The first is the requirement to generate data off-line. This requires very long run times (in excess of 8 hours) on the CRAY computer. The second is the lack of a true closed loop operation. The ASSE will overcome these problems and add new features shown in Figure 3.14 to increase the capability.

The first major step is a parallel implementation of the OSC tape or the FASTSIG version. This will give a fast, direct way to generate data without resorting to off-line supercomputers. Performance will be higher and turn-around time much quicker.

The second major step is to revise the computational procedure to permit true closed-loop operation. When this is done new features can be added to more accurately represent the seeker, objects and background. This will also provide multi-spectral capability for a single simulation run which will be important in testing more robust seekers and signal processors.

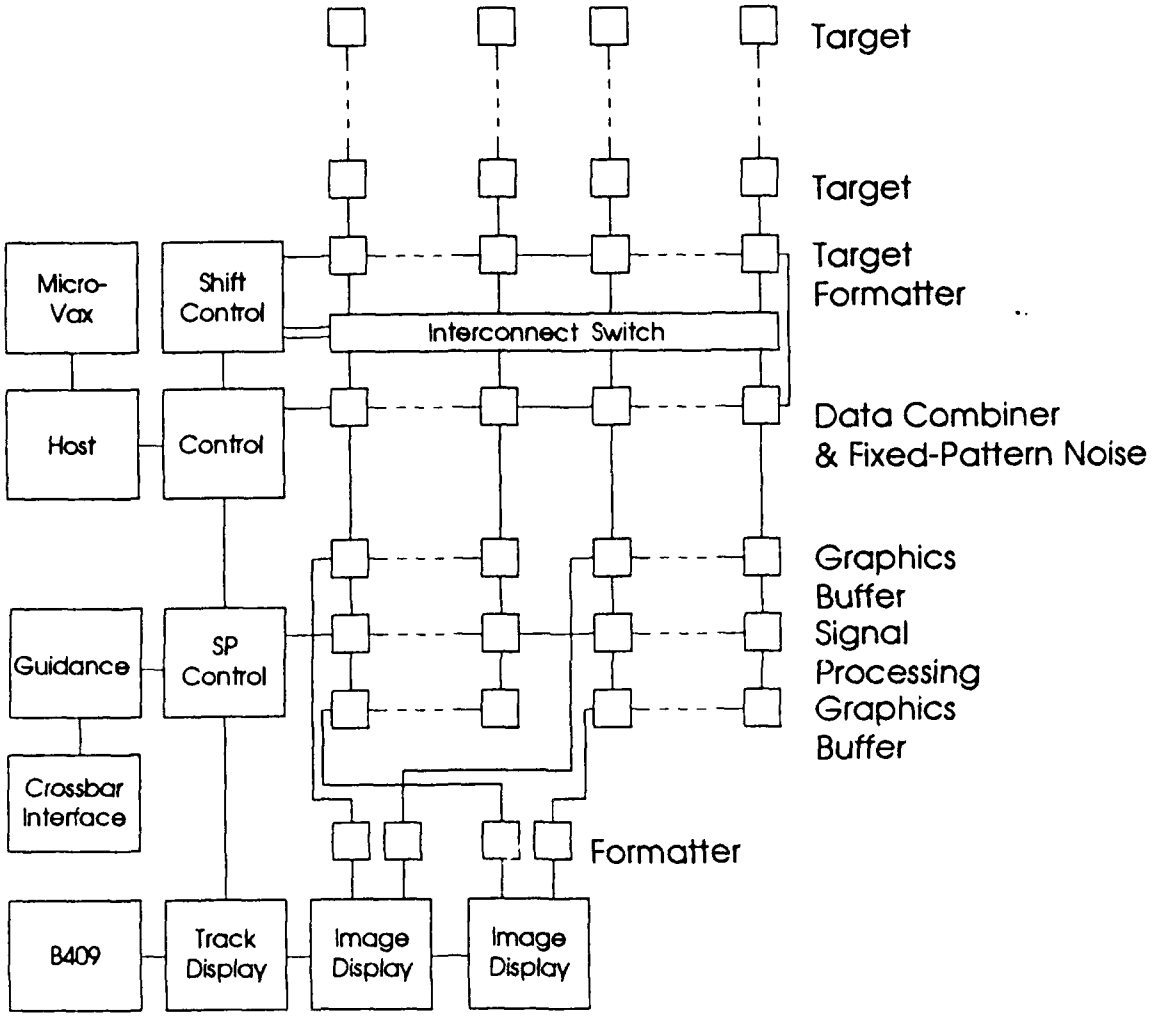


FIGURE 3.11 SEEKER/SCENE EMULATOR SYSTEM

MG - \WIN30\GORDON\FR1990/SSE-SYS.DRW

- Custom 16-bit interface to Ga. Tech Parallel Function Processor
  - Conforms to Inmos Transputer Module specification
  - Conforms to Ga. Tech PFP Crossbar Interface specification

- Custom Power and System Services backplane

- Multi-Transputer graphics system
  - 1280 x 960 resolution
  - 256 colors
  - Up to 90 Hz operation
  - Simultaneous update of image by multiple processors
- IBM PC-AT Host
- DEC Microvax II File Server
  - 4 Gigabyte Tape Drive
  - Transputer Interface (Caplin Cybernetics LTD.)

Figure 3.12 SSE Peripherals and Interfaces

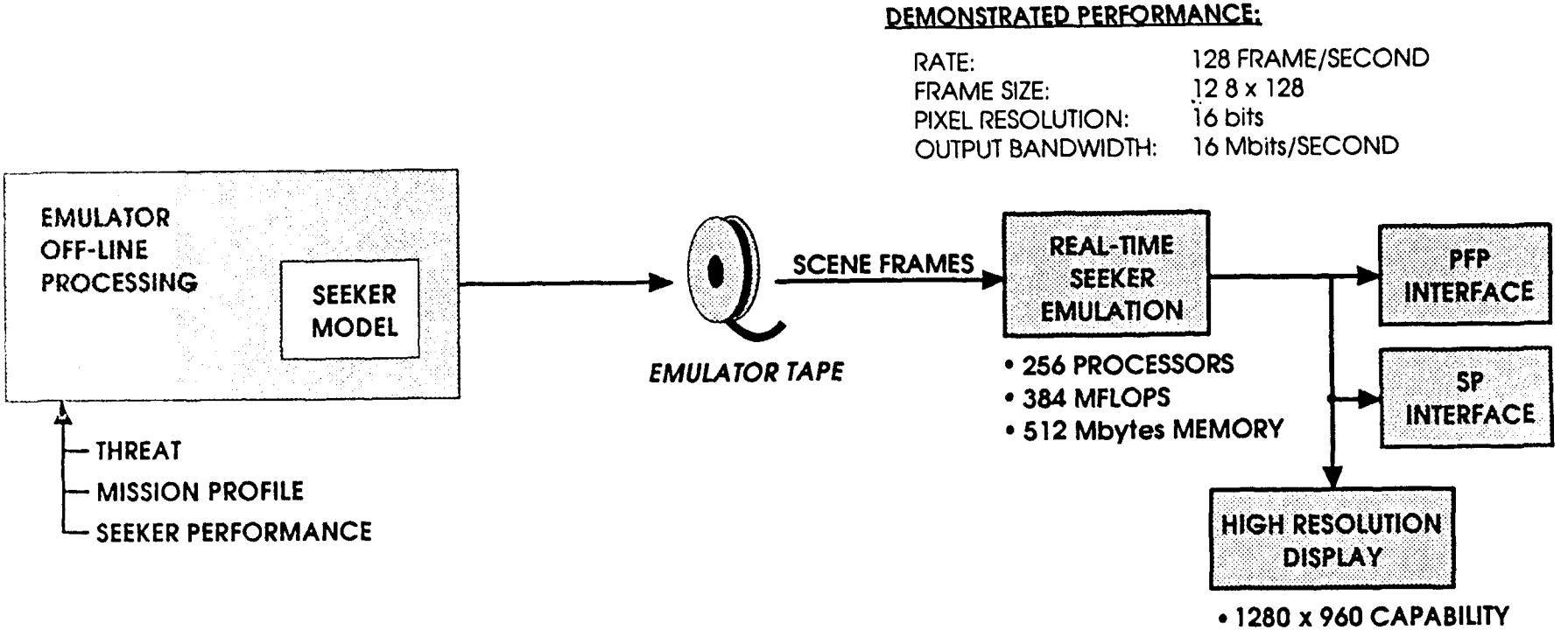


FIGURE 3.13 SSE REAL-TIME SYSTEM DEMONSTRATION

MG - \WIN30\GORDON\FR1990/SSERTSD.DRW

- Parallelization of off-line code for processing by Transputers
- Advanced Seeker Scene Emulator
  - Wide FOV, Closed-Loop Operation
  - Increased Fidelity Nuclear Effects Modeling
  - Multi-spectral Capability
  - Complex Threat Geometry and Dynamics
  - Dynamic Tailoring/Selection of Scene Scenario
  - LATS Seeker Model Anchored to LETS Test Results

Figure 3.14 Advanced Seeker/Scene Emulator

4. GUIDANCE, NAVIGATION & CONTROL PROCESSOR

4.1. Objectives

A review of GN&C Processor requirements for KEW interceptors led to the following objectives for this task:

Modular, expandable design

VLSI chip designs based on a Silicon Compiler

Clear path to VHSIC technology and to rad-hard VLSI technology

Capability to support KEW mission over the next 20 years

These objectives were translated into a specific processor design composed of three processor types; signal processor, data processor and executive processor. The signal processor was further divided into pixel processing and object processing. Objectives for each processing mode were set as shown in Figure 4.1. While a single color FPA is implied by these objectives, the modular structure makes the addition of more sensors relatively straightforward. This will become apparent in Section 4.2.

4.2. *Capabilities*

The GN&C processor developed by Georgia tech is a parallel processor architecture that provides direct support for application of the functional processing concept to the guidance, navigation, and control of Kinetic Energy Weapons (KEW) interceptors. Based on the computing requirements for the guidance, navigation, and control of KEW interceptors, the GN&C processor is functionally decomposed into three general classes of processor architectures: data processor (GT-DP), signal processor (GT-SP), and executive processor (GT-EP). A fully-connected 8-point crossbar switch is used to connect the various processor modules in a closely coupled interconnection network. Each processing module is tailored to the unique computational requirements of each functional block. The result is a parallel processing system with a computational throughput that meets the most stringent KEW requirements. The architecture of the GN&C processor and its capabilities are presented in the following sections.

4.2.1. Architecture

A modular, parallel architecture was chosen to meet the processing requirements. The architecture, shown in Figure 4.2, provides a processing throughput range of 9:1. Processors, of differing types, can be added to the system from a minimum of one up to a maximum of 9. In addition the signal processor can be selected or not selected. If selected the minimum configuration is one processor plus the signal processor.

PIXEL PROCESSING CAPABILITY

- SUPPORT 100 Hz, 128 x 128 FPA PIXEL DATA RATE
- PROVIDE FOR UP TO 5 POINT NON-UNIFORMITY COMPENSATION PER PIXEL
- PROVIDE FOR UP TO 4 POLE/4 ZERO TEMPORAL FILTERING
- PROVIDE FOR UP TO 9 POINT/7 COEFFICIENT SPATIAL FILTERING
- VARIABLE MODE THRESHOLDING (FIXED AND ALGORITHMIC)
- NON-PROCESSOR LIMITED CENTROIDING CAPABILITY (HUNDREDS OF NON-CONTIGUOUS OBJECTS)

OBJECT PROCESSING CAPABILITY

- PERFORM MULTIPLE OBJECT PROCESSING IN PARALLEL
- 8-10 MFLOPS PER OBJECT PROCESSOR AT 32 BIT FLOATING POINT
- UP TO 8 OBJECT PROCESSORS TOTALING 80 MFLOP LIMIT (TRADE WITH DP)

DATA PROCESSING CAPABILITY

- UP TO 8 PROCESSORS (TRADE WITH OP)
- 6 MFLOPS PER DATA PROCESSOR AT 32 BIT FLOATING POINT

EXECUTIVE PROCESSING CAPABILITY

- FLEXIBLE INTERRUPT WITH RESPONSE WITHIN 400 nsec
- MULTIPLE CHANNEL I/O INTERFACE UP TO 16 BIDIRECTIONAL, 32 BIT CHANNELS
- 8-10 MFLOP EXECUTIVE CONTROL AND INTERFACE HANDLING PROCESSING CAPABILITY

FIGURE 4.1 GEORGIA TECH GN&C PROCESSOR OBJECTIVES

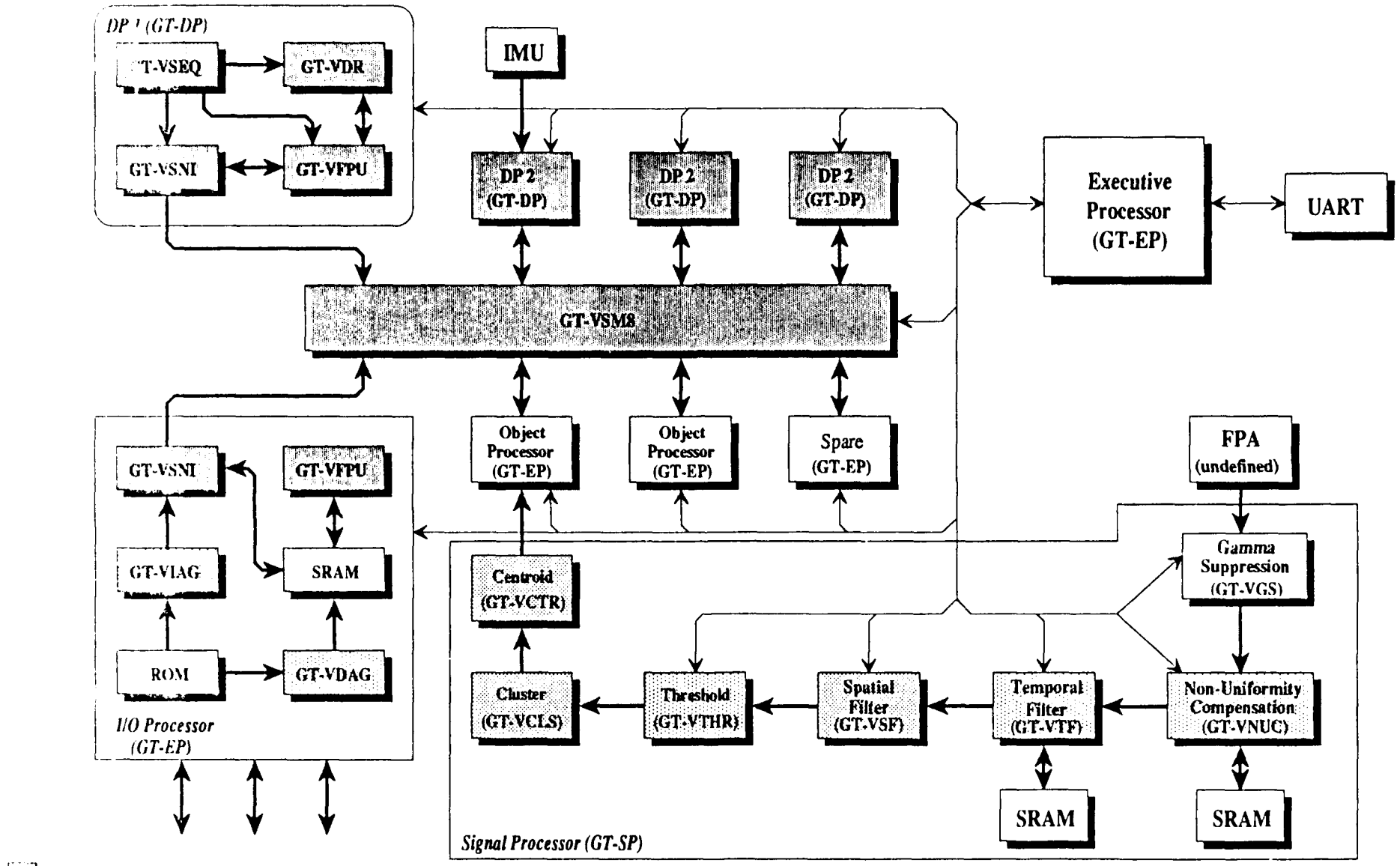


FIGURE 4.2 GN&C PROCESSOR ARCHITECTURE

4.2.1.1. Data Processor: GT-DP

The data processor is used to perform numerically intensive tasks for guidance, navigation, and control of the KEW interceptor. This type of computation is floating-point intensive and requires very high scalar throughput. These computational tasks do not require large amounts of instruction and data memory (less than 1 k bytes). The Georgia Tech GT-DP processor was designed to meet these requirements. Four GT-DP processors are shown in Figure 4.2, but this number can be increased or decreased to meet the GN&C data processing needs.

As shown in Figure 4.3 the GT-DP processor consists of four functional blocks: Instruction Control, Data Control, Arithmetic Control, and Communication Control. The Instruction Control Unit (GT-VSEQ) is responsible for the generation of instruction addresses. It receives status flags from the Arithmetic Control Unit and appropriately determines the next instruction addresses. It also facilitates branch-lookahead for efficient pipeline arithmetic instruction execution.

In each computing cycle, the Data Control Unit (GT-VDR) supplies two operands to the Arithmetic control unit. In addition, it receives a result form the Arithmetic Control Unit for storage. Three addressing modes are supported: direct, indexing, and post-indexing. The Arithmetic Control Unit is used to perform the actual data computation. Three data types are supported: floating-point, fixed-point, and bit-field. The Arithmetic Control Unit operates in three pipeline stages: 1 stage for operand fetches, 1 stage for data computation, and 1 stage for a result store. An automatic operand-dependency scheme to control the internal feedback paths in the Arithmetic Control Unit, and branch-look-ahead facility in the Instruction Control Unit, enable the GT-DP processor to execute scalar computations efficiently.

The Communication Control Unit designated as the GT-VSNI is used to control the communication between the GT-DP processor and other processor modules connected to an 8-point fully-connected network. The GT-VSNI chip consists of two pairs of 32-level deep FIFOs. One pair of FIFOs is used to communicate with other processors through the crossbar network. Another pair is used to directly communicate with the executive processor. An 8-point fully-connected crossbar switch enables multiple data processors to effectively exchange data and state variables. The switching matrix chip designated as the GT-VSM8 is designed to directly interface with the GT-VSNI chip. It provides 8 input ports and 8 output ports to connect up to 8 GT-VSNI chips. All processor modules communicate with the network through the GT-VSNI chip.

4.2.1.2. Executive Processor: GT-EP

The executive processor provides overall executive control for the GN&C processor. Among the tasks to be executed by the executive processor are initialization of the GT-DP and GT-SP processors,

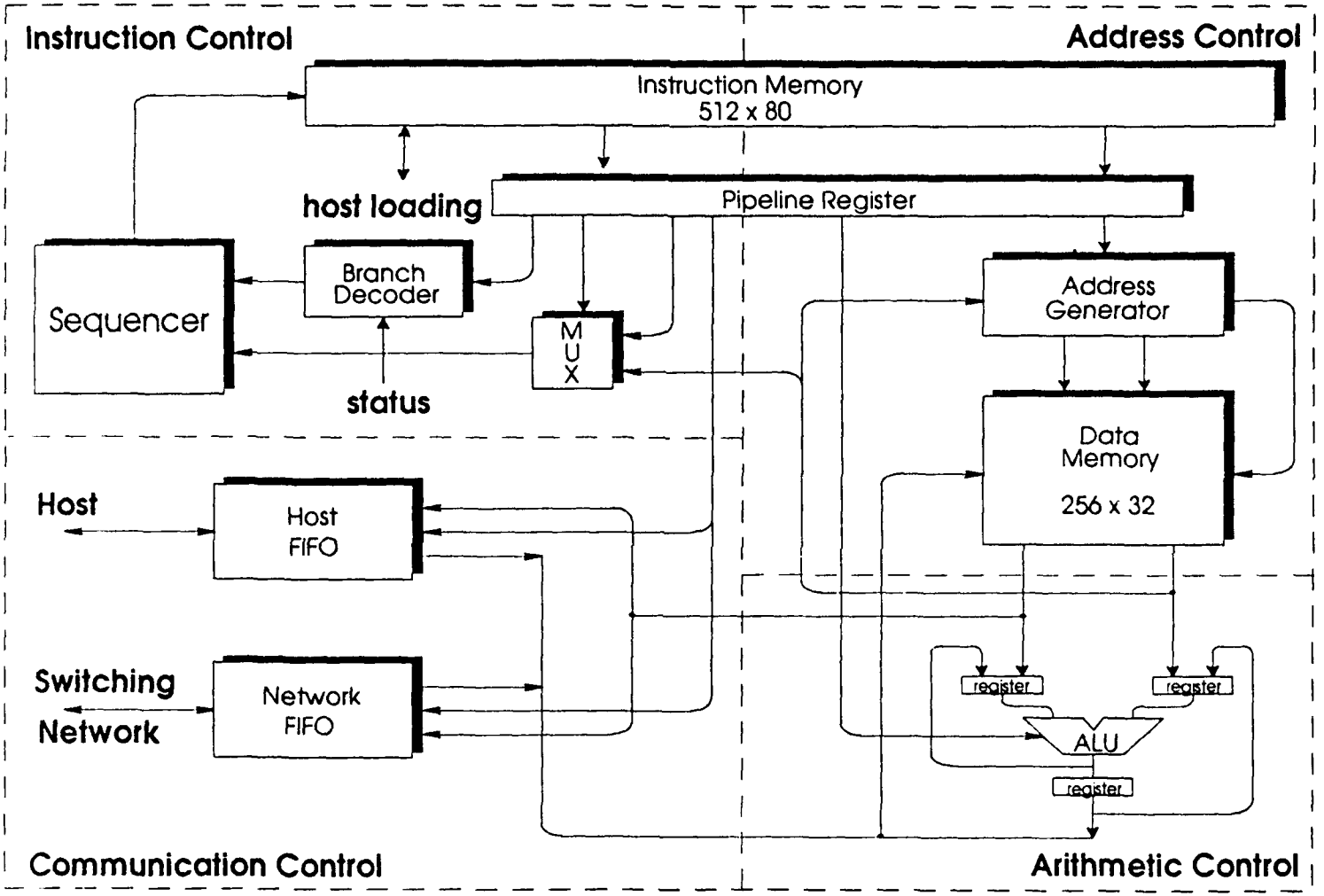


FIGURE 4.3 GT-DP PROCESSOR ARCHITECTURE

MG - \WIN30\GORDON\FR1990/DP-ARCH.DRW

overall system consistency checks, flight phase/mode control, target tracking, and interfacing with other devices such as the IMU and control valves. To perform these executive functions, the GT-EP processor needs to have access to considerably larger amounts of instruction and data memory than the GT-DP processor. In addition, the GT-EP processor must handle real-time tasks and event scheduling in which fast interrupt response capability is critical. Furthermore, the GT-EP is designed to meet the Object Processing requirements. A total of 5 GT-EP processors (shown in Figure 4.2) are used on the Georgia Tech GN&C processor system: 1 as the executive processor, 1 as an I/O processor, and 3 as Object Processor.

As shown in Figure 4.4, the GT-EP processor consists of six functional units: instruction memory, data memory, instruction address generation, data address generation, arithmetic logic unit, and network interface. The arithmetic logic unit uses the GT-VFPU chip developed for the GT-DP processor. The network interface uses the GT-VSNI chip developed for the GT-DP processor.

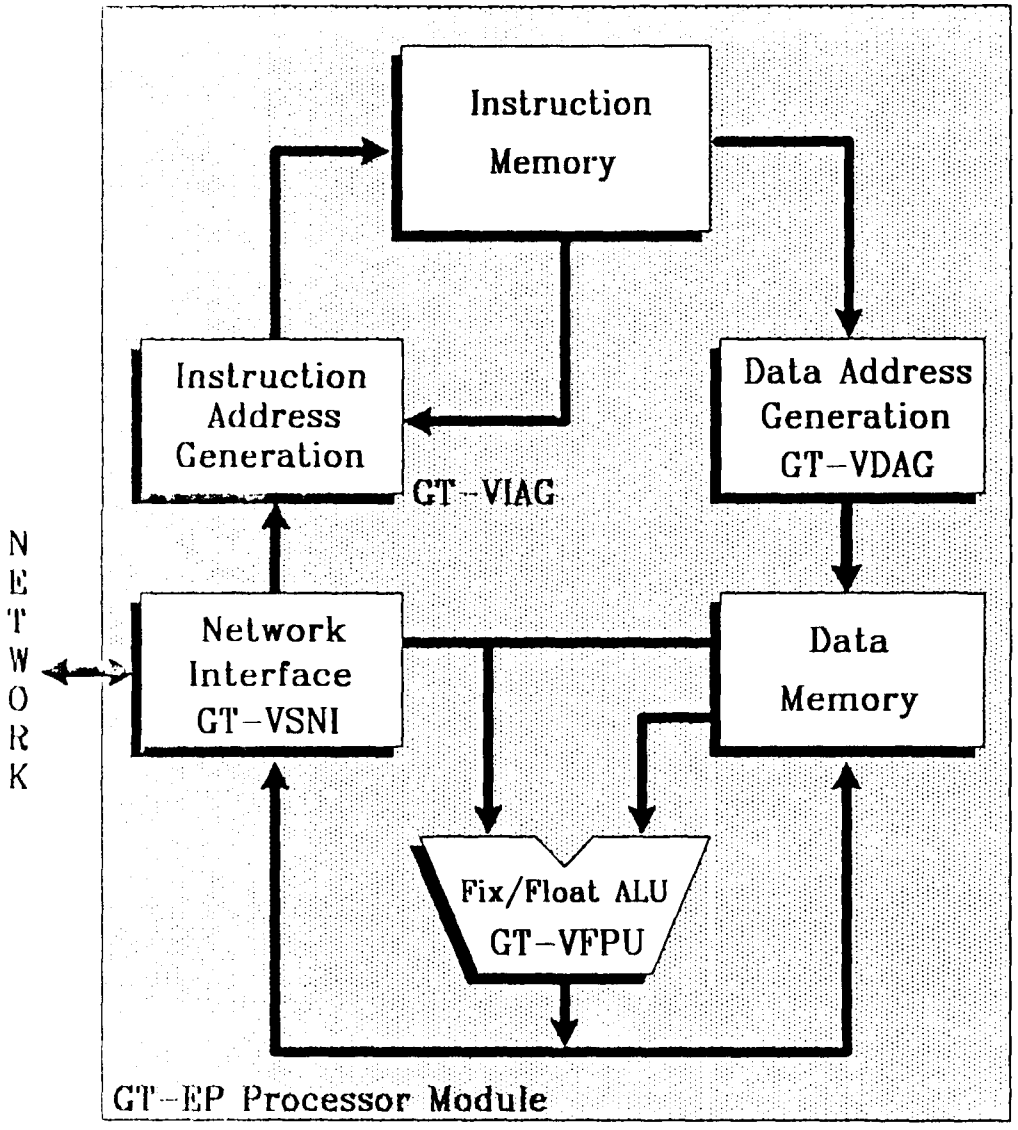
The primary function of the GT-VIAG chip is to generate addresses for the instruction memory. In addition, it provides an opcode field to control signals to the GT-VFPU fixed/floating point arithmetic logic unit. For I/O functions, the GT-EP processor can directly access 16 input and output devices. Four of the 16 channels are reserved for asynchronous devices. The other 12 channels are used with synchronous devices.

The GT-VDAG chip is used to generate two address fields for data/operand fetches and one address field for data/result store. The chip supports post-index addressing for accessing arrays with constant strides at a rate of one cycle per element of the array. Relative addressing is supported to ease the access of local variables and parameters of procedures that are recursive. An automatic operand-dependency check is used in the GT-VDAG to alleviate the need to insert *nops* at the end of every basic program block associated with a three-stage pipelined ALU such as the GT-VFPU chip. The GT-VDAG chip supports a data space of 64 MW.

4.2.1.3. Signal Processor : GT-SP

The signal processor was developed to process infrared images for a focal plane array (FPA) with 128 x 128 pixel resolution at a rate of 100 frames per second. Each pixel is assumed to have 12-bit resolution with a dynamic range of 16 bits. The signal processor performs various forms of filtering operations on the pixel data before clustering them into objects for target tracking and discrimination. The signal processor is decomposed into 8 functional blocks for VLSI implementation. The first functional block is the FPA interface (GT-VFPA) which is used to link the signal processor and the FPA. The FPA has not been specified. As a result this functional block is not defined.

The second block is the GT-VGS which is used to address problems associated with gamma suppression. In order to filter out gamma spikes, the GT-VGS needs to process pixel data from the FPA at a very high frame rate (10,000 frames per second). Gamma spikes are represented by pixels that



GT-EP Processor Features :

- 32 Bit Data Word
- 25M SP Whetstones Performance
  (Calculated based on 15M Whetstones Prototype)
- 16 Channels I/O Capability
- Expandable to
  64M Words Data Memory
  64M Words Instruction Memory
- 16 Real-time Interrupts
- Two 32-bit Built-in Timers
- Supports Ada Language
- Self-test Capability
- Supports Real-time Emulation and Control Capabilities
- Supports PFP & Flight Hardware

FIGURE 4.4 GT-EP PROCESSOR ARCHITECTURE

MG - \WIN30\GORDON\FR1990/EP-ARCH.DRW

exceed a certain threshold. Pixels that fall below the gamma spike threshold are accumulated. Pixels that exceed the gamma spike threshold are suppressed (value set to zero). At a lower frame rate (100 frames per second), each pixel is represented by the accumulated pixel value compensated by adding the number of pixels exceeding the gamma spike threshold multiplied by the average value of the gamma-spike-free pixels.

The third functional block, GT-VNUC, is used to compensate nonlinear detector characteristics in the FPA. The response of each detector is compensated with 4 piecewise linear segments. During calibration, the FPA is irradiated with five known sources. As illustrated in Figure 4.5, based on the FPA response, 4 linear segments are constructed for each pixel. During normal operation, each pixel value is mapped from one of the four linear segments to a common desired response.

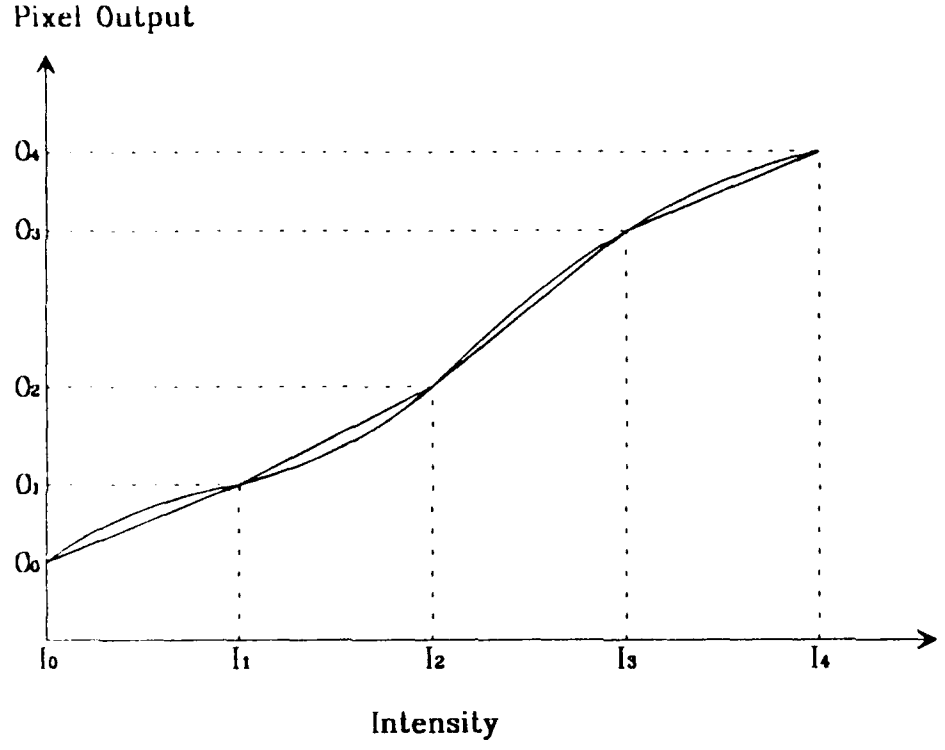
The fourth functional block is the GT-VTF which performs time averaging of pixel values across frames. The GT-VTF is used reduce random noise across frames as well as smearing of images due to a jittering motion on the FPA. The GT-VTF is implemented as a forth order temporal filter that makes use of pixel values from the previous four frames. Eight coefficients in the GT-VTF can be programmed from a host port to achieve a desired filter response. Figure 4.6 illustrates the filtering operation of the GT-VTF.

The fifth functional block is the GT-VSF 9-point spatial filter. The GT-VSF performs filtering operations based on the pixel value and that of its immediate eight surrounding pixels. The GT-VSF can be used to reduce the effects of spatial noise as well as to enhance/reduce the contrast of images. Figure 4.7 illustrates the functionality of the GT-VSF.

The sixth functional block, GT-VTHR, is used to suppress noise by cutting out pixels that exceed a constant or calculated threshold. Three types of thresholding are supported: simple, adjusted, and adaptive. Simple thresholding uses a constant lower threshold and fixed upper threshold set by the host. Adjusted thresholding allows the lower threshold value to be dynamically adjusted according to the number of pixels passed by the GT-VTHR on the previous frame. Finally, adaptive thresholding computes the lower threshold based on a statistical average of the 8 pixels which surround the pixel under evaluation. The three thresholding modes of the GT-VTHR are illustrated in Figure 4.8.

The seventh functional block is GT-VCLS. As illustrated in Figure 4.9, the GT-VCLS groups adjacent pixels with non-zero intensity into clusters. Two non-zero pixels are assigned to a cluster if the distance between them is no more than 1. The diagonal distance between two pixels is considered a 1. Each cluster is tagged and merged with other clusters when two pixels, one from each cluster, touch.

The last functional block of the signal processor is the GT-VCTR centroiding chip. The statistical information of each cluster from GT-VCLS is computed by the GT-VCTR. As shown in Figure 4.10, the total intensity, the intensity centroid, the area A(total number of pixels), and the area



For each pixel:

Calibration:

- Apply 5 known sources: I_n, n = [0,4]
- *Sample* Output *responses*: O_n, n = [0,4]
- Obtain 4 piecewise Linear functions:

$$PW(O_r, n) = \frac{(I_{n+1} - I_n)}{(O_{n+1} - O_n)}(O_r - O_n) + I_n$$

for n = [0,3]

Compensation :

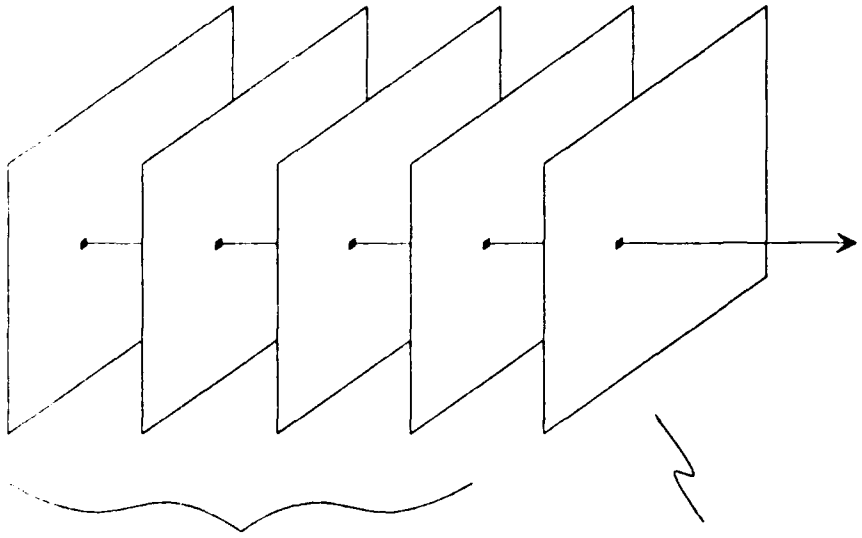
- Sample output response: O_r
- Find n such that $O_n < O_r < O_{n+1}$
- Calculate compensated intensity:

$$I_c = PW\ (O_r, n)$$

- Storage Requirement (5*128*128*16) = 1.3 Mbits
- Computational Requirement (2+1+1+4)*128*128*100 = 13 MOPS

FIGURE 4.5 GT-VNUC NON-UNIFORMITY COMPENSATION

MG - \WIN30\GORDON\FR1990/SP-OVER.DRW



Four Previous Data Frames

Current Data Frame
128 x 128 x 16 bits

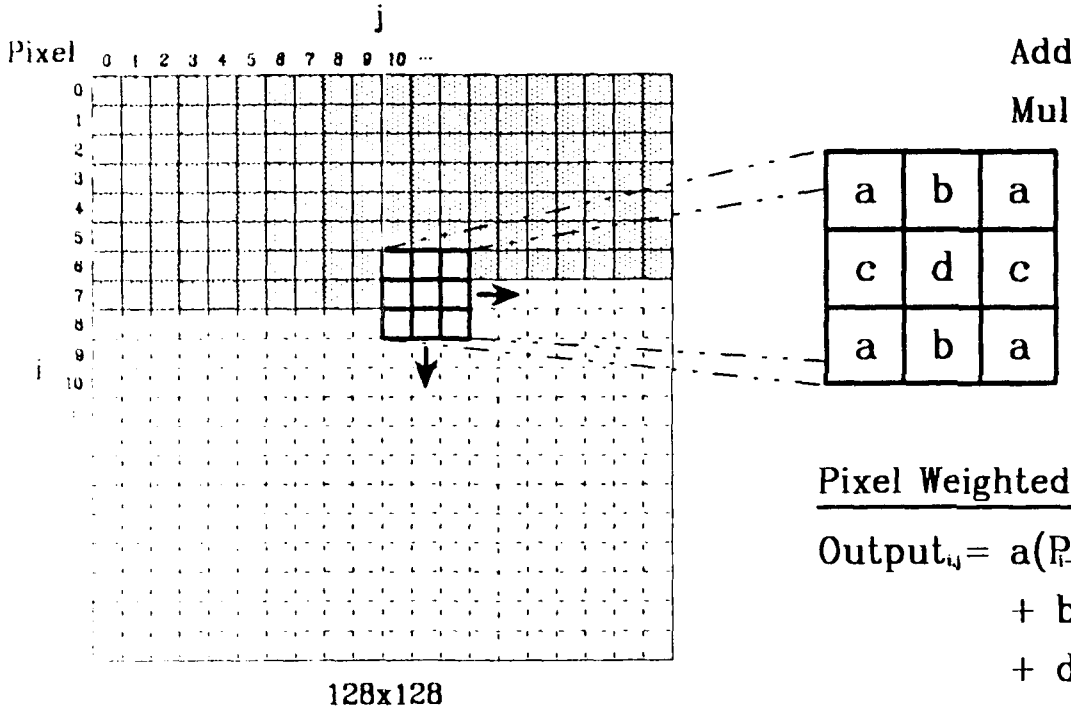
$$\text{Pixel } P_{ij}(t_n) = \frac{\sum_{K=0}^{4} b_K z^{n-K} P_{ij}}{1 + \sum_{K=1}^{4} a_K z^{n-K} P_{ij}}$$

- Storage Requirements: 4*128*128*16 bits = 1 Mbits
- Add Operations (100 frames/sec): 8*128*128*100 = 13 MOPS
- Multiply Operations: 7*128*128*100 = 11 MOPS

24 MOPS

FIGURE 4.6 GT-VTF TEMPORAL FILTERING

9 Point, 2D, FIR Filter



Processing Requirement (100 frames/sec)

Add Operations : 8*128*128*100 = 13 MOPS

Multiply Operations : 4*128*128*100 = 7 MOPS

20 MOPS

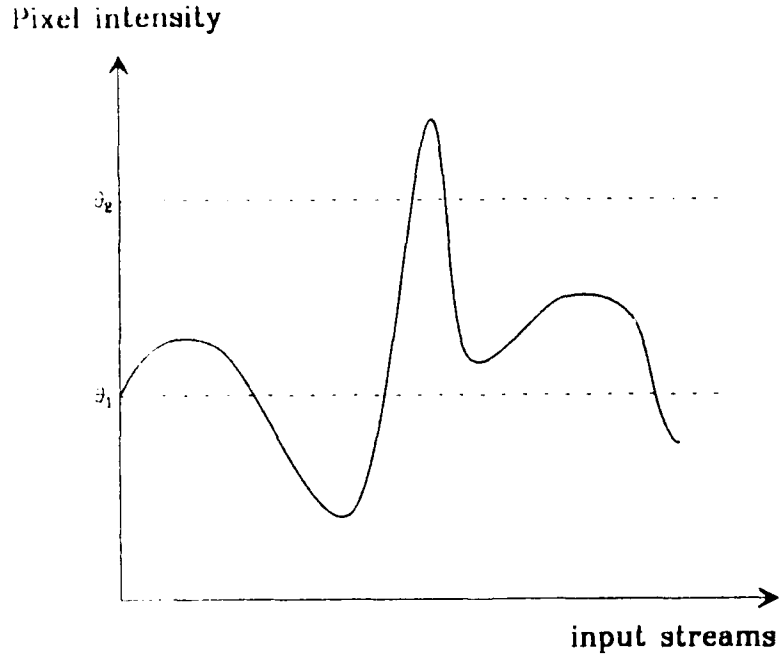
Pixel Weighted Sum

$$Output_{ij} = a(P_{i-1,j-1} + P_{i-1,j+1} + P_{i+1,j-1} + P_{i+1,j+1}) + b(P_{i-1,j} + P_{i+1,j}) + c(P_{i,j-1} + P_{i,j+1}) + d(P_{i,j})$$

Scanned across each row and column

FIGURE 4.7 GT-VSF SPATIAL FILTERING

MG - \WIN30\GORDON\FR1990/SP-OVER.DRW



$$O_{ij} = \begin{cases} I_{ij}, \ \forall\ \theta_1 \leqslant I_{ij} \leqslant \theta_2 \\ \theta, \text{ otherwise} \end{cases}$$

1. Simple thresholding: θ_1 & θ_2 constants
2. Adjusted thresholding

   θ_2 : constant

   $\theta_1 = Z^{-1}\theta_1$ op delta

   $$\text{op} \begin{cases} +, & \#\text{ pixels} > N_1 \\ -, & \#\text{ pixels} < N_2 \\ \text{nop}, & N_1 \leqslant \#\text{ pixels} \leqslant N_2 \end{cases}$$

   Z^{-1}: frame delay

   # pixels: total number of pixels in a frame

   N_1, N_2: constants controlled by the executive

3. Adaptive thresholding (53 MOPS)

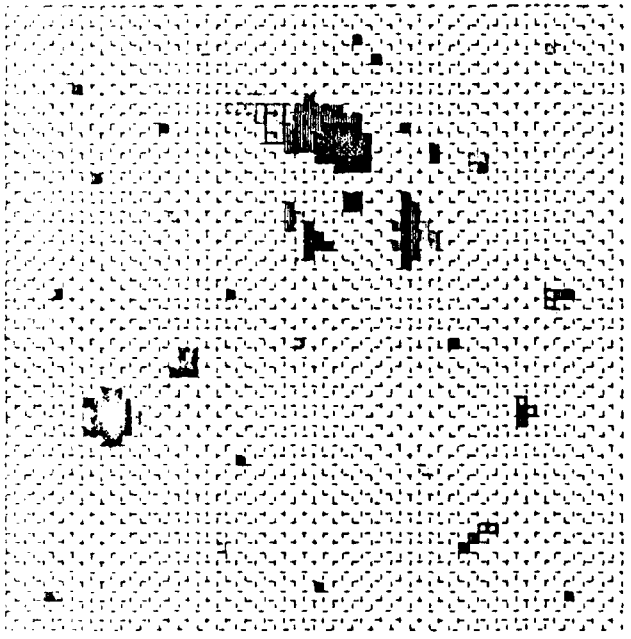
   ϑ_2 : constant

   $\vartheta_1 = k_1 * E + k_2 L_1 + k_3$; $k_1 \geqslant \theta$

   E_1 = average value over neighboring 8 pixels (8*128*128*100=13 MOPS)

   L_1 = absolute sum of the difference between E and neighboring 8 pixels (23*128*128*100=37 MOPS)

FIGURE 4.8 GT-VTHR THRESHOLDING



- Pixel P(i,j) and P(k,l) are adjacent if

  $(|i-k| < 2)$ & $(|j-l| < 2)$ & $[(j \neq k) \,||\, (j \neq l)]$

- Adjacent pixels with non-zero pixel intensity are grouped into a cluster

- Pixels are scanned from left to right and top to bottom

- A cluster is completed if no pixel in a row is adjacent to any pixels in the cluster

- Maximum number of clusters in a 128x128 FPA is 64x64 = **4096**

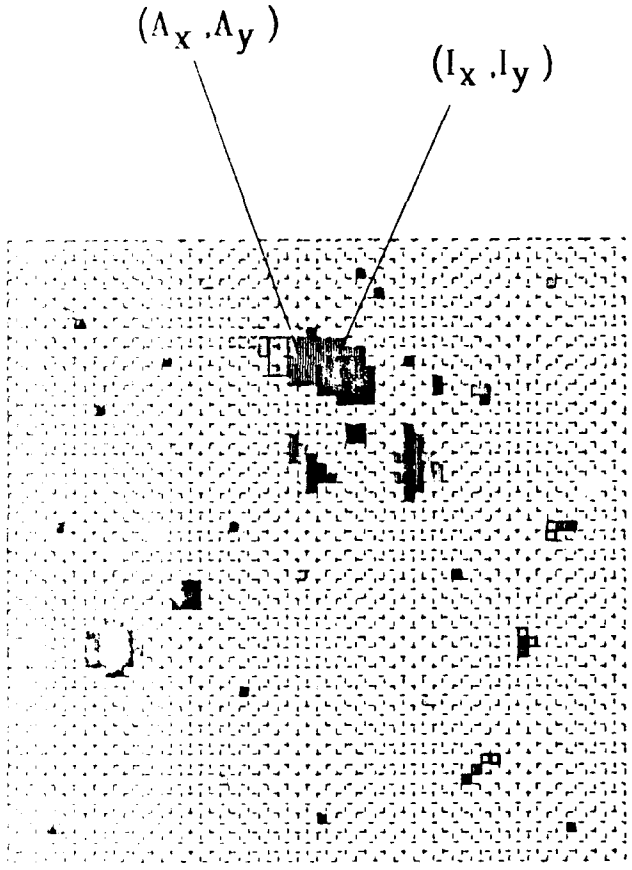
- Parallel algorithm for the identification and grouping of clusters requires 128 entry associative search

- Total equivalent processing requirement =

  493 MOPS

FIGURE 4.9 GT-VCLS CLUSTERING

MG - \WIN30\GORDON\FR1990/CLUSTER.DRW



- For each cluster in the field of view calculate :

Area : #A = total number of pixels in the cluster

Intensity : #I = Sum of all intensity values of each pixel in the cluster

Area Centroid : For all i and j such that pixels P(i,j) is an element of the cluster,

$$A_x = \frac{\sum x(i,j)}{\#A}$$

$$A_y = \frac{\sum y(i,j)}{\#A}$$

Intensity Centroid: For all i and j such pixel P(i,j) is an element of the cluster,

$$I_x = \frac{\sum I(i,j)\ x(i,j)}{\#I}$$

$$I_y = \frac{\sum I(i,j)\ y(i,j)}{\#I}$$

Computational requirement : 25 MOPS

FIGURE 4.10 GT-VCTR CENTROIDING

MG - \WIN30\GORDON\FR1990/CENTROID.DRW

centroid of each cluster are calculated. Each finished cluster is sent to the object processor for target acquisition, tracking, and discrimination.

4.2.2. Performance

The operating and physical characteristics of the Georgia Tech GN&C processor are based on VLSI designs using the Genesil silicon compiler. The designs utilize the National Semiconductor 1.25 micron CMOS process and the Hewlett Packard 1.0 micron CMOS process.

Each GT-DP processor consists of 4 VLSI chips. The die size of each chip is shown in Figure 4.11. At an operating speed of 6.6 Mhz and a power consumption of 5.68 Watts, each GT-DP processor is capable of executing 6.6 MFLOPS. Using General Electric High-Density Interconnect (HDI) hybrid packaging technology, the GT-DP processor can be packaged in a 1" x 1" x 0.2" space [39]. Assuming that the hybrid board has an equivalent weight density of 4 ounces per square foot copper foil, each GT-DP processor would weigh approximately 80 grams.

The GT-DP processor contains two I/O ports. One of the I/O ports is used for direct communication with the executive processor and has a bandwidth of 40 Megabits per second (Mbps). The second I/O port has a bandwidth of 40 Mbps and is used for parallel processing applications. The two I/O ports combined, provide an I/O bandwidth of 80 Mbps with a total of 36 I/O pins for the GT-DP processor.

The GT-EP executive processor consists of four VLSI chips (GT-VIAG, GT-VDAG, GT-VFPU, and GT-VSNI) and the necessary memory chips for the instruction and data memories. The number of chips required for the instruction memory depends on the amount of instruction and data memory used with the GT-EP processor. In a typical configuration with 8k of instruction and data memory using off-the-shelf 8kx8 memory chips, the number of memory chips required is 8 for the data memory and 11 for the instruction memory. Assuming that each memory chip is 280 x 280 mil^2, using the GE HDI packaging technology, 8 memory dies can be placed on a single 1" x 1" 0.2" hybrid circuit board. A second board can be used to hold another nine memory chips. The GT-VIAG and GT-VDAG VLSI chips are approximately 400 x 400 mil^2. The two chips and an additional memory die can be packaged on a third hybrid board. The total package size for the GT-EP executive processor with 8k of instruction memory and 8k of data memory will be 1" x 1" x 0.8". Each memory chip consumes about 0.15 Watts and each of the two VLSI chips consumes approximately 1.0 Watts. The total power consumption for the GT-EP package is 7.967 Watts with a performance throughput of 10 MFLOPS. The I/O bandwidth of theGT-EP processor is 640 Mbps. The package weighs approximately 320 grams. The characteristics of the GT-EP processor are summarized in Figure 4.12.

Chip	Size (Mil2)	Power (Watts)
GT-VFPU GT-VDR GT-VSEQ GT-VSNI	362x383 501x524 373x423 276x301	2.5 1.85 0.718 0.617
Total	642,025	5.68
Package	1"x1"x0.2", 80 grams / GT-DP	
Performance	6.6 MFLOPS	80 Mbps

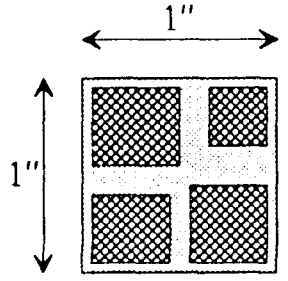


FIGURE 4.11 CHARACTERISTICS OF THE GT-DP PROCESSOR

MG - \WIN30\GORDON\FR1990\DP-CHAR.DRW

Chip	Size (Mil2)	Power (Watts)	#required
GT-VIAG	400x400	1.0	1
GT-VDAG	400x400	1.0	1
GT-VFPU	362x383	2.5	1
GT-VSNI	276x301	0.617	1
Memory (8kx8)	280x280	0.150	19
Total	1,809,600	7.967	23
Package	1"x1"x0.8", 320 grams / GT-EP		
Performance	10 MFLOPS	640 Mbps	152 kB

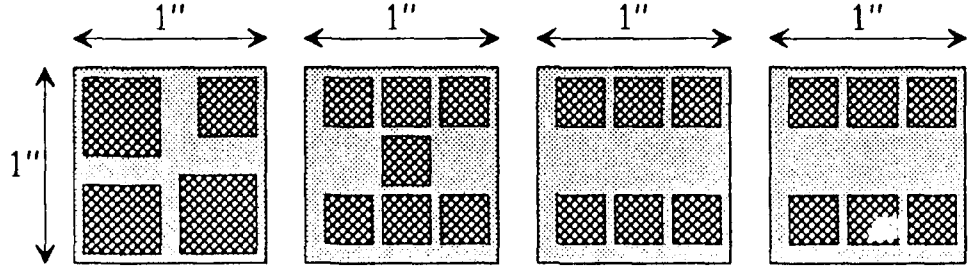


FIGURE 4.12 CHARACTERISTICS OF THE GT-EP PROCESSOR

MG - \WIN30\GORDON\FR1990/EP-CHAR.DRW

The GT-SP processor requires 7 custon VLSI chips and 36 off-the-shelf 8k x 8 memory chips. The physical and operating charactistics of the chips are shown in Figure 4.13. Figure 4.14 shows the packaging loyout of processor. The number of MOPS in Figure 4.13 is based on a frame rate of 10,000 for the GT-VGS gamma suppression chip and 100 for the others. The GT-VSF, GT-VTHR, GT-VCLS, and GT-VCTR chips are capable of operationg at 200 frames per second which effectively doubles the MOPS figures for the chips. The total throughput for the Georgia Tech GT-VSP is in excess of 969 MOPS. The only support chips required for processor are memory chips for GT-VNUC and GT-VTF. The GT-VNUC requires twenty 8k x 8 RAM chips and the GT-VTF requires sixteen 8 x 8 RAM chips. Each of the RAM chips consumes approximately 0.15 Watts. The total power consumption for processor is 28.6 Watts. The front-end of the GT-SP processor has sixteen GT-VGS chips which process pixel data at 10,000 frames per second. At this rate each GT-VGS chip provides a bandwidth of 163.8 Mbps with a total 2,261 Mbps for the 16 GT-VGS chips. The back-end of the GT-SP processor has two GT-VCTR chips which provide cluster information for the object processor. The clustering chips are capable of handling a maximum of 64 x 64 = 4096 clusters. Each cluster contains 78 bits of information: area (20 bits), x-coordinate centroid (7 bits), y-coordinate centroid (7 bits), intensity (30 bits), x-intensity centroid (7 bits), and y-intensity centroid (7 bits). At 100 frames per second, the back-end of processor has an I/O bandwidth of 100 x 64 x 64 x 78 = 32 Mbps.

The GT-VGS chips can be packaged in four 1" x 1" x 0.2" hybrid boards. The GT-VNUC with its supporting memory chips requires three boards. The 36 memory chips required for the GT-VNUC and GT-VTF can be packaged in four boards. The GT-VNUC, GT-VTF, GT-VTHR, and GT-VSF can be packaged in one hybrid board. Lastly, the GT-VCLS and two GT-VCTR chips can be packaged in another hybrid board. The total number of hybrid boards required for the GT-SP processor is ten with a combined board space of 1" x 1" x 2".

The GT-VSM8 is an 8 x 8 crossbar network with 8 input ports and 8 output ports for data transfer between the various processing units as shown in Figure 4.2. Each port has a bandwidth of 20 Mbps for a total of 320 Mbps for the GT-VSM8 chip. The GT-VSM8 has a die size of 329 x 340 mil^2 and consumes 0.8 Watts of power. It can be packaged in a 1"x1"x0.2" board. The characteristics of the network chip are shown in Figure 4.15.

The parallel processor architecture shown in Figure 4.2 uses four GT-DP processors, five GT-EP processors (1 as the executive processor, 1 as the I/O processor, and 3 as object processors), and one GT-SP processor. The characteristics of the GN&C processor system are shown in Figure 4.16. The GN&C system occupies a space of 1"x1"x7" with a power consumption of 91.92 Watts and weighs approximately 2800 grams. The computing power of the GN&C is 76.4 MFLOPS for data and object processing functions. A computing power of 973 MOPS is provided for signal processing. The system I/O bandwidth is 3.8 Gbps (giga bits per second) for data/object processing and 2.6 Gbps for signal processing.

With a progression to 0.5 micron CMOS technology, the Georgia Tech GN&C processor system can be packaged in 1" x 1" x 2". With a factor of 2 improvement in speed the GN&C processor would then have a performance fo 152 MFLOPS for data/object processing and 1946 MOPS for signal processing with an expected I/O bandwidth of 12.3 Gbps. The projected characteristics of the GN&C processor with 0.5 micron technology are shown in Figure 4.17.

Chip	MOPS	Size (Mil2)	Power (Watts)	#required
GT-VGS	20	400x400	1.0	16
GT-VNUC	12	400x400	1.2	1
GT-VTF	23	400x400	1.0	1
GT-VSF	18	378x401	0.8	1
GT-VTHR	53	415x415	1.1	1
GT-VCLS	493	370x370	0.9	1
GT-VCTR	25	395x395	1.1	2
Memory (8kx8)	N/A	280x280	0.15	36
Total	969	4,078,313	28.6	59
Package	1"x1"x2", 800 grams / GT-SP			
Performance	969 MOPS 2,653 Mbps			

FIGURE 4.13 CHARACTERISTICS OF THE GT-SP PROCESSOR

MG - \WIN30\GORDON\FR1990/SP-CHAR.DRW

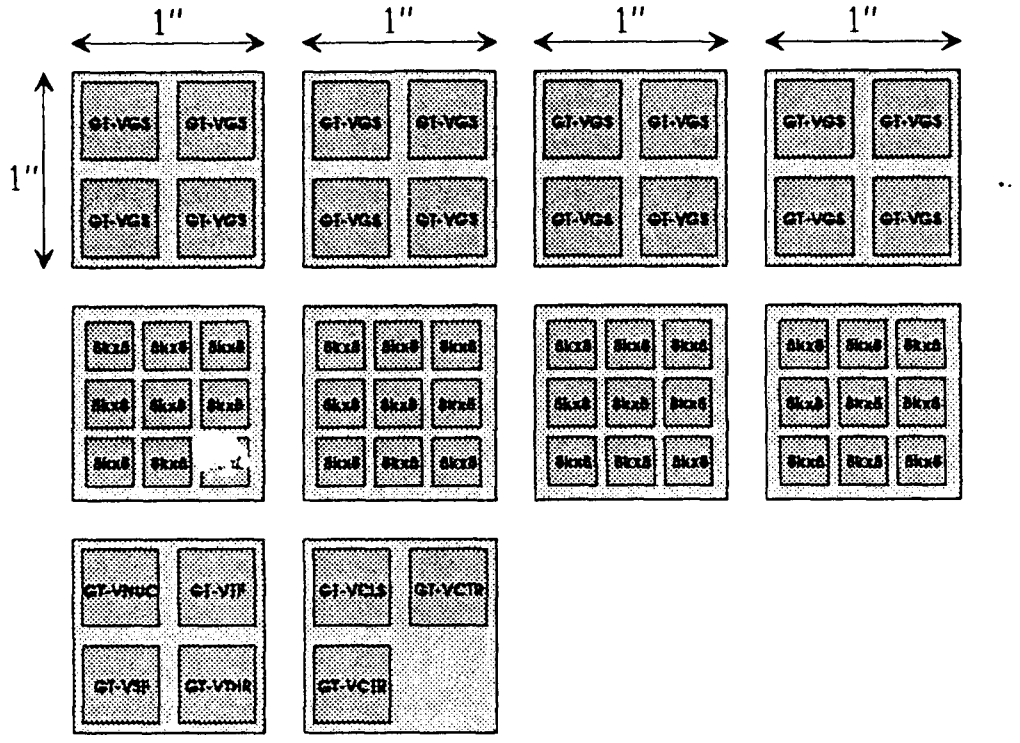


FIGURE 4.14 GT-SP PROCESSOR PACKAGING

MG - \WIN30\GORDON\FR1990/SP-CHAR.DRW

Chip	Size (Mil2)	Power (Watts)
GT-VSM8	329x340	0.8
Package	1"x1"x0.2", 80 grams / GT-DP	
Performance	320 Mbps	

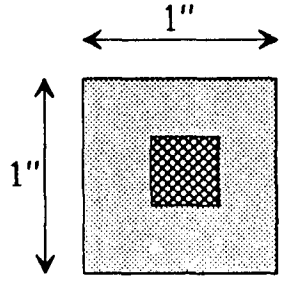


FIGURE 4.15 CHARACTERISTICS OF THE GT-VSM8 CROSSBAR NETWORK

MG - \WIN30\GORDON\FR1990/SM8-CHAR.DRW

Chip	Pakage	Power (Watts)	#required
GT-DP	1"x1"x0.2"	5.68	4
GT-EP	1"x1"x0.8"	7.96	5
GT-SP	1"x1"x2.0"	28.60	1
GT-VSM8	1"x1"x0.2	0.80	1
Total	1"x1"x7"	91.92	11
Package	1"x1"x7", 2800 grams		
Performance	76 MFLOPS, 973 MOPS, 6493 Mbps		

FIGURE 4.16 CHARACTERISTICS OF THE GT-GN&C PROCESSOR

MG - \WIN30\GORDON\FR1990/GNC-CHAR.DRW

Chip	Pakage	Power (Watts)	#required
GT-DP GT-EP GT-SP GT-VSM8	1"x1"x0.2" 1"x1"x0.2" 1"x1"x0.6" 1"x1"x0.2"	8 3 13.8 0.8	1 5 1 1
Total	1"x1"x2"	37.6	8
Package	1"x1"x2", 800 grams		
Performance	152 MFLOPS, 1946 MOPS, 13586 Mbps		

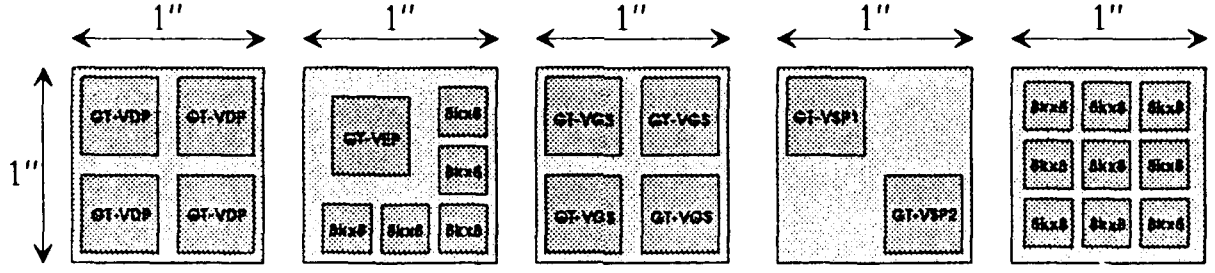


FIGURE 4.17 CHARACTERISTICS OF THE GT-GN&C PROCESSOR WITH 0.5u TECHNOLOGY

MG - \WIN30\GORDON\FR1990/GN&CTECH.DRW

5. SOFTWARE DEVELOPMENT

5.1. Objectives

The parallel hardware developed under the program will be virtually worthless without user friendly software development tools. Software is needed to support the development of 6 DOF simulations for KEW interceptors, flight software for the GN&C Processor and Seeker/Scene models for the SSE. All of this software needs to be Ada compatible. Georgia Tech has initiated a software development program that addresses this issue.

5.2. Capabilities

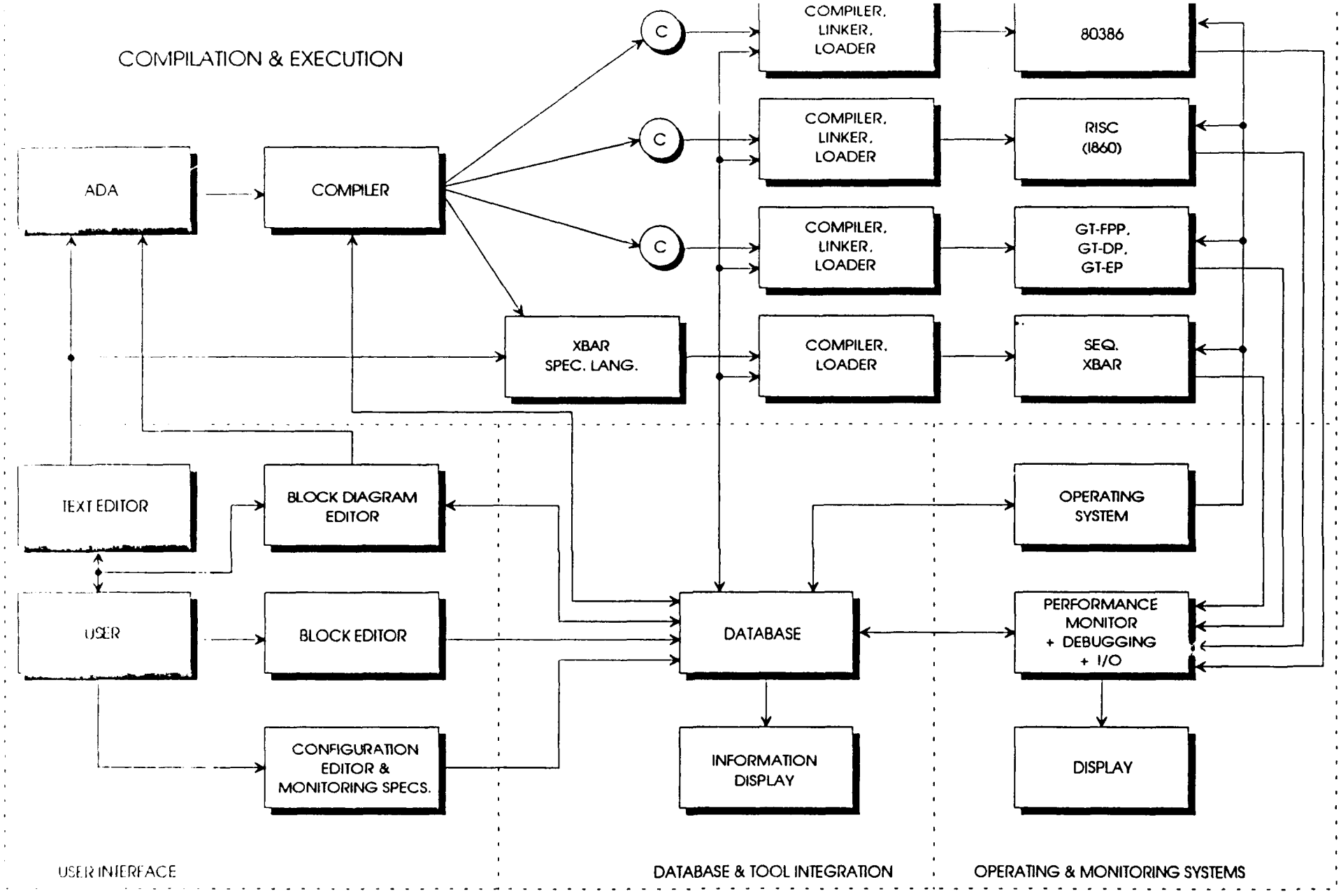
5.2.1. Overview of Software Architecture

Georgia Tech is devising an integrated parallel programming framework (IPPF) for the development of software for the special purpose parallel processor architectures. The IPPF serves as an entity for the integration of diverse hardware components and provides a consistent interface for effective exploitation of hardware capabilities. The IPPF consists of four components: user interface, compilation and execution, operating and monitoring system, and database and tool integration. The overall software architecture of the IPPF is shown in Figure 5.1.

5.2.1.1. User Interface

The conventional programming method uses direct text editing of program source code. This mode of programming is still supported in the IPPF environment. However, the primary method of programming is through a block editor and block diagram editor. Using the block editor, the user creates and defines basic functional programming blocks. Each functional block is represented by a graphical block diagram. Data flowing into and out of the block are represented by input and output connection ports. The behavior of the block is represented by a self-contained code segment with receive commands for information flowing into the block and send commands for information flowing out of the block. The block diagram editor allows a user to assemble predefined functional blocks and connect the blocks into a higher level system of functional blocks. Using this method, a complex application program can be graphically and hierarchically developed.

The configuration editor and monitoring specification allows a user to effectively control hardware resources and specify specific monitoring information for an application run. The default hardware configuration is automatically extracted from the application block diagram. Manual configuration is only needed if optimization around a particular hardware configuration is desired.



A FULL SPECTRUM OF SOFTWARE DEVELOPMENT AND APPLICATION TOOLS IS BEING IMPLEMENTED TO SUPPORT PFP USE FOR REAL-TIME TESTING

FIGURE 5.1 INTEGRATED PARALLEL PROGRAMING FRAMEWORK

MG - \WIN30\GORDON\FR1990\IPPFWRK.DRW

The text editor, block editor, block diagram editor, configuration editor, and monitor specification are incorporated into an integrated mouse-controlled, menu-driven, graphical environment.

5.2.1.2. Compilation and Execution

Regardless of the form of user interface, either through block diagram editing or direct program editing, the eventual output from the user interface is program source code. The source code is directed to an appropriate compiler to generate the target object code for execution. The programming languages supported are Ada, Pascal, C, and FORTRAN. A crossbar compiler is used to compile the specification of a communication configuration into interconnection patterns between processing elements. Each processor type requires a separate compiler, linker, and loader. Processor specific low level utilities are incorporated into the environment database.

5.2.1.3. Operating and Monitoring System

The operating system provides efficient run-time system constructs for effective utilization of the underlying capability of hardware resources. It provides a concise interface between the various programming languages (Ada, C, Assembly language, etch.) and the underlying architecture of each type of processor element. It also provides a facility for exception handling and error recovery mechanisms. The operating system gives each processor element the ability to execute multiple tasks. This is especially important when the number of functional blocks exceeds the number of available processor elements. The monitoring system provides a useful mechanism to obtain feedback from an application run. It provides capability for a systematic display of application results and a structured facility for real-time data collection. The monitoring system also serves as the interface for system debugging.

5.2.1.4. Database and Tool Integration

The software database provides a repository for the block diagrams, configuration information, monitoring specification, language-specific package library, target-machine specific utilities, operating system constructs, and monitoring systems. The database serves as an integration tool for the other three components of the IPPF. A database interface provides constructs to create, access, update, and display the information in the database.

5.2.2. Characteristics of Parallel Programming Environments

Highly parallel architectures offer opportunities for significant improvements in program execution speed and reliability. However, these opportunities cannot be realized unless effective program development tools are available. A parallel programming environment (PPE) differs from conventional program development systems in several ways.

5.2.2.1. Explicit and Implicit Parallelism

A program's parallelism must be expressed (1) explicitly by the programmer and/or (2) implicitly as part of a PPF's "compilation" of program source. Typically, larger to medium grain parallelism may be expressed explicitly. The effectiveness of implicit parallelism has been demonstrated for smaller grain parallelism and for specific application domains (e.g., for functional descriptions of real-time simulations).

For a PP, (1) and (2) imply that explicitly expressed parallelism should be visible and easily modifiable by applications programmers, whereas implicit parallelism may be visible but need not be directly modifiable. For the real-time simulations addressed by the PFP project, we will use domain-specific representations of parallelism, in the form of graphical descriptions of functional decompositions.

5.2.2.2. Parallelism and Target Parallel Machines

Typically, programming models have to be specialized for specific target parallel machines and operating systems. For example, for many hypercube applications, synchronous styles of programming have been shown effective, whereas asynchronous (or chaotic) algorithms have been highly successful on non-uniform memory multiprocessors. Therefore, any PP that attempts to provide support for parallel programming needs to be able to support multiple backend parallel machines and it must also support the use of multiple programming models. For the PFP multicomputer, a synchronous style of programming appears appropriate. In addition, a PP must be able to use standard programming languages, such as Ada and C.

5.2.2.3. Parallelism and Application Domains

For the explicit expression of parallelism, the programming model presented to the programmer should address the specific properties of the programmer's application domain. For example, in real-time simulations, low-level control functions are easily described as statically decomposed collections of communicating functional blocks. This suggests that a useful programming model is one that presents the functional (possibly replicated) building blocks in the application using graphical descriptions. This is the approach we will pursue for the PP being constructed for the PFP, in conjunction with visual illustrations of the performance effects of such decompositions.

More importantly, one of the environment's attributes will be its ability to exploit application domain-specific knowledge for assistance in parallel programming. For example, when performing resource allocation for PFP's real-time simulations (e.g., mapping functional blocks to processors), the system will use built-in mapping functions, thereby removing from programmers the responsibility of computing and enforcing such mappings.

5.2.2.4. Performance Evaluation and Improvement

Since the primary objective of parallel computing is performance improvement, a PP must assist the programmer in gaining understanding of program performance on the target parallel machine. This implies (1) that tools for program monitoring, performance evaluation or prediction and for the visualization of performance information should be integral parts of the programming system and (2) that programmers should be assisted in making changes to their parallel applications in response to such evaluations or predictions - termed program tuning. We will address this issue in the context of the abstract information representation described next. Specifically, we will assume that the PP, should provide a general framework that makes effective use of a wide variety of performance display, evaluation, and visualization tools.

5.2.2.5. Abstract Information Representation

Whether programs are evaluated and tuned by inspection and alteration of their parallel structure, of resource allocation decisions, or of specific program components, the representation and manipulation of the program required for making such changes should be straightforward. This requires that a PP maintain abstract representations of the executable version of the parallel program and of its execution environment that are easily inspected and manipulated and contain sufficient information for program generation and alteration, as well as for the visualization and evaluation of its performance. This abstract representation must describe parallelism in a fashion that is "neutral" with respect to the program representation at routine and the program description at compile-time. The representation developed as part of this research is termed program views. It describes a parallel program at two levels: (1) at the high level as a set of interacting and related entities with certain attributes using a data model similar to the entity-relationship database model and (2) at a lower level as a set of interacting processes, where process interactions are described as messages sent and received on abstract communication channels. In addition, since multiple components of tools in a PP must jointly access or manipulate the information repository(ies) associated with the information model, the model is defined such that the repository(ies) offers the following functionality: (1) facilities for sharing information between cooperating tools, (2) facilities for tool control via the shared repository, and (3) facilities for tool observation. Topics (1)-(3) are discussed below.

5.2.2.6 Operating Software and Parallel Machines

Since higher performance parallel applications must make efficient use of the target system's hardware, routine software actually consists of both application code and operating system code, and program tuning also concerns the performance of the operating system components used by the application. Specifically, tuning may concern operating system configurations, such as the selection among alternative operating system components, the determination of relevant OS parameters (e.g., setting buffer six for message communication), or even the synthesis or construction of appropriate OS

components in conjunction with the generation of executable program code. This implies that the proposed PP cannot be constructed with the assumption that all parallelism is mapped to a single set of routine constructs. Instead, the "operating system" actually consists of a variety of routine primitives configurable by the application programmer.

5.2.3 Technologies for Parallel Programming Environments

The successful development of a PP requires the following technologies:

Language and compiler technology for the support of multiple models of parallel programming (when explicitly describing parallelism) and for the automatic or semi-automatic parallelization of programs. Here, we are using and developing Ada and C compilers, linkers, and loaders, enhanced with libraries and special-purpose packages for inter-processor communication and coordination on the target parallel machine.

For program generation, analysis, and improvement: programming environment, database, and visualization technology for the representation, sharing, and display of information about the parallel program, its execution environment, and its run-time performance. This is the PP being developed in this research.

Performance analysis, program specification techniques, and, perhaps, Artificial Intelligence technology (1) for performance modeling of the parallel program, (2) for expression of relevant performance attributes of parallel programs or of program invariants not to be changed during performance tuning, and (3) for relating models and measurements to actual program code as well as for suggesting and making changes to such code. Here, we will offer simple means of performance evaluation and program visualization, coupled to a graphical interface used for program development.

Operating system technology for efficient performance monitoring and for the efficient execution of parallel programs on different target parallel machines. Here, we have developed a configurable and portable operating system kernel presenting a "library" of primitives used by the programmer. Portability is essential in light of the multi-CPU nature of any PFP, machine, which may contain both special purpose processors (such as the FPP) and general-purpose processors like the Intel 386 or 860 boards.

It is beyond the scope of a single research effort to develop novel technologies in all of the areas listed above. Therefore, for purposes of this work, we are focusing on operating systems and programming environment technology:

Compiler technology and environment tools. Due to the special-purpose nature of the target hardware, we have been designing and implementing a variety of tools for hardware use, including low-level device drivers, system monitoring and configuration software, compilers and assemblers for the

target parallel machine, loaders, linkers, etch. In addition, significant efforts have been expended on facilitating system installation, by use of Unix Makefiles. The environment is structured as a collection of tools sharing a common information store, called the abstract information representation.

Abstract information representation and tool integration. As a prerequisite for the support of multiple models of parallel programs, we have designed and implemented a high level information model and associated information repository that can be used to represent relevant program information (compile-time and routine information, information about the program and its execution environment). In addition, we have designed the model so that it and the repository may be used with multiple tools for information sharing, mutual observation, and mutual control.

Performance evaluation, improvement (tuning), and visualization. We have prototyped a graphical user interface to the system and are now using our experiences with this prototype to implement a framework for the graphical construction of parallel programs and for the visualization of their performance.

Operating system technology. We have developed a configurable kernel that has been ported to both the special-purpose processors (FPP) on the PFP, as well as the general-purpose CPU's (currently, the Intel 386). Kernel communication primitives offer substantially increased functionality compared to that offered by the basic hardware, yet are quite competitive in performance.

Future work may concern the use of graphical techniques for program tuning and an investigation of the role of precise specification of program functionality or performance (e.g., performance models) in the tuning of program performance.

5.3 Current Status of the Software Development

The current implementation of the Integrated Parallel Programming Framework (IPPF) consists of four primary tool sets: Parallel Program Construction System (PPCS), Parallel Program Monitoring System (PPMS), Parallel Program Tuning System (PPTS), and Tool Integration System (TIS).

A programmer uses the PPCS for the initial development of a parallel application. When the program is sufficiently complete, the programmer runs it and gathers performance data with the PPMS. Based on the performance measurements, more development with the PPCS might be necessary or the PPTS can be used to tune the completed application for the target execution environment. These three tool sets are supported by and coordinated by the TIS.

The PPCS encompasses all tools used during program development: a graphical user interface, recompilation drivers, compilers, assemblers, program libraries, linkers, loaders. Currently, the PFP PPCS has a prototype graphical user interface (called "bde" or "Block Diagram Editor") which runs under

the X windows system, a sophisticated system of Makefiles and shell scripts which drive the recompilation process, The Makefiles and scripts not only drive the recompilation of application programs but also the recompilation and installation of the PFP environment itself. A compiler which translates a subset of Ada to C, a C compiler for the FPP, assemblers for both the FPP and FPX, a linker for the FPP, and loaders for the FPP, FPX and Intel 386 processors. These tools were all designed and implemented in-house. Additionally, there are vendor-supplied compilers, assemblers, and linkers for the Intel 386 processor and vendor-supplied and publicly available compiler tools and text editors for the Sun host environment. Much work has been done in supporting the FPP: the C compiler, under development for over a year and a half, has been subjected to a C test suite (the so-called "C Torture Test") and appears to be stable. The compiler is currently being targeted to the FPX processor. We have purchased a commercial Ada to C translator to replace our Ada-subset compiler and we are beginning to integrate it with the rest of the IPPF. This is part of a validated Ada compiler from Irvine Compiler Corporation.

The PPMS design is very new. Other than this design work, a rudimentary implementation exists for verifying timing characteristics of real-time simulations, but it does not yet provide any hints on where performance problems exist within an application or how it might be modified to increase its performance. A less rudimentary implementation which indicates communication bottlenecks is underway.

The PPTS currently consists primarily of a compiler which produces process-to-processor mappings and interprocessor communication patterns. The language for this compiler and the compiler itself are currently being redesigned. The PPTS will ultimately include a graphical user interface where a programmer can easily reconfigure and tune his parallel program for its target execution environment.

The TIS provides the basis for the PPCS, the PPMS, and the PPTS to work together effectively and efficiently. It consists of several tools: a number of device drivers to control the PFP hardware, a library supporting host program interaction with the hardware and the parallel programs it is executing, and an automatic hardware configuration tool. The configuration tool also serves as an effective hardware testing and diagnostic device, and a newly-completed entity-relationship-set database system which implements the abstract information representation described above. Currently, each of these tools is implemented. We are in the process of refitting some of the pieces of the PPCS, PPMS,and PPTS to use the database system.

The current PFP environment and the associated PFP configurable kernel has supported the development of the following applications: Satellite Attitude Control Simulation, 3-DOF Missile Simulation (Terminal Phase), Multiple Threat/KEW Interceptor Simulation.

The implementation of the operating kernel is detailed in the PFP Kernel Design Document which is submitted under a separate technical report. The structure of the environment database is

described in the Shared Persistent Data Structures design document which is also submitted under a separate technical report.

5.4 EXOSIM

5.4.1 Overview

During the contract period, BDM Corporation was used as a subcontractor to develop a 6-DOF simulation of an Advanced Exoatmospheric Interceptor. Their initial Simulation was KWEST. This work was turned over to Coleman Research Corporation as a continuing effort. Coleman was funded under a separate contract to complete the simulation with the objective of a parallel real-time implementation on the Georgia Tech PFP.

The following paragraphs are taken (with modification) from the Coleman documents delivered to USASDC [40 - 43].

The development of EXOSIM was initiated in October, 1988, using ERIS Baseline models and subroutines. The BOOST-phase simulation, denoted KEERIS Version 1.0 and written in ACSL/FORTRAN, was delivered to KEW on February 1, 1989 and verified against results of an equivalent LMSC simulation.

A conversion of the simulation to all-FORTRAN was completed in mid-May. Extensive enhancements of the KV-phase models, including autopilot, Kalman filter, and the ACS/VCS logic were incorporated as well as integration of a staring Focal Plane Array (FPA) seeker noise model form the BDM Corporation. In addition, in order to achieve a true end-to-end simulation, POST-BOOST and MID-COURSE flight modes were simulated at a low level of fidelity. These improvements resulted in EXOSIM Version 1.0, delivered to KW on 30 June 1989.

The development of several additional enhancements was initiated in July, 1989, and completed for EXOSIM Version 2.0. The primary upgrades were : incorporation of an IMU dynamic response model, a high-fidelity seeker model, signal/object processing algorithms, and a detailed midcourse autopilot model. In addition, several changes were incorporated into the simulation structure to enhance partitioning onto the PFP.

5.4.2 Simulation Capabilities

5.4.2.1 Inertial Measurement Unit (IMU)

The IMU model in EXOSIM Version 1.0 was a conventional strapdown unit run with truth states and instantaneous (ideal) time response. For EXOSIM Version 2.0, second-order response characteristics were incorporated. Error models for non-orthogonality, misalignment, scale factor, bias, and drift terms

were not explicitly defined in software for the Resonant Fiber Optic Gyro (RFOG) and Quartz Resonant Accelerometer (QRA) components of the SSIMU. The approach taken by CRC was to retain the flexibility available in the generic IMU model, currently resident in EXOSIM, and adapt this model through data input and subsidiary code to create a SSIMU-specific simulation in future upgrades.

5.4.2.2 Midcourse Guidance and Attitude Control

The midcourse guidance in EXOSIM version 1.0 used a separate subroutine to control KV pitchover, and midcourse event sequencing, including divert burn scheduling. For EXOSIM Version 2.0, a midcourse guidance routine and a midcourse autopilot were incorporated to control midcourse sequencing and KV orientation. Immediately following boost, the roll rate is driven to zero and the KV is pitched over. When the estimated attitude errors and body rates are below the midcourse autopilot thresholds, the seeker shroud, nose fairing, and booster adaptor are dropped and four VCS disturbance burns are sequenced. The angular acceleration induced by thruster misalignments is recorded, and used by the KV autopilot for disturbance compensation. The KV is then rolled to align the nearest VCS thruster with the perpendicular component of the velocity to be gained vector, VG, and the first midcourse divert occurs. A second divert occurs approximately half-way between the first divert and the range at which seeker acquisition is anticipated. The final divert occurs just prior to anticipated acquisition.

5.4.2.3 High Fidelity Staring FPA Seeker Model

EXOSIM Version 1.0 employed a simple, single target angle noise model provided by the BDM Corporation. For EXOSIM Version 2.0, a CRC-developed detailed (pixel-level) 128 x 128 array model (CRC Seeker Model) was added in order to implement elementary, multiple object discrimination and single target tracking and guidance algorithms for assessment of Kalman filter performance. The model uses the LATS optical telescope design and focal plane size parameters but is specific to a 8-14 micron MCT array with MIS architecture and CID parallel readout via shift register clocked charge preamplifiers. Although the preamp frequency shaping networks are designed for a maximum frame rate of 128 per second, the model is currently running at 100 per second to be compatible with the GIT PFP and seeker emulator interface.

Noise sources modeled include: (a) optical (background or "Photon" noise), (b) preamplifer, consisting of JFET preamplifier white and 1/f current and voltage noises, Johnson noise in the input and feedback resistors, and shot noise due to leakage/dark current, (c) KTC (reset) noise, (d) Fixed pattern noise due to parasitic capacitances and (e) non-uniformity effects in both the detectors and preamplifiers. The 1/f noise in the detectors, a phenomena which is not well understood, and sampling (aliasing) noise, are not modeled at present.

The model can simulate up to ten point targets of varying radiant intensities and a fifty-point resolved target for terminal phase imaging and aimpoint selection. In the model used for EXOSIM Version 2.0, a benign complex of five objects with arbitrary signatures is assumed to allow cursory simulation of signal and object processing algorithms, which are briefly described in Sections 5.4.2.4 and 5.4.2.5. The characteristics and types of objects selected, while representative, bear no relation to tactical requirements. The threat composition, shown in Table 5.1, has been selected to insure discrimination and target designation effectiveness.

Table 5.1 Objects Contained in Seeker FOV at Acquisition

OBJECT NO	TYPE	RADIANT INTENSITY (W/STER)
	TARGET	5.4.2.3.18.CSO (UNRESOLVED @ $R_{acquition}$)
	ETA - BALLOON	5.4.2.3.19.CSO (UNRESOLVED @ $R_{acquition}$)
	TANK BODY	
	RHO - BALLOON	
	DEPLOYMENT DEBRIS/FRAGMENT	

The on-focal-plane (analog) processing for the LATS seeker is still under study by LMSC and its subcontractors. For EXOSIM Version 2.0, several analog processing functions have been assumed. These include: background substration and pedestal clamping, signal limiting, and AGC (frame rate select). The method of nonuniformity compensation is not defined: a residual (uncompensated) value of .05 percent is assumed for both detectors and preamps.

5.4.2.4 Signal Processing

Simplified signal processing algorithms were developed by CRC for the acquisition (signal to noise ratio, thresholding and pixel false alarm reduction, function. In order to reduce computer run time, full-field signal processing is performed on the first frame only, which is the only one in which all of the 128 x 128 pixels are sampled. This technique allows selection of only one object or cluster for subsequent tracking and object processing, and the only pixels sampled subsequent to target selection are those in the immediate vicinity of the tracked object. All other pixels are gated out, which reduces the simulation time on a VAX 11/750 from approximately five minutes per frame to less than three seconds per frame, allowing the KV phase guidance simulation to be run on a frame-by-frame basis all the way form designation to impact or guidance termination.

5.4.2.5 Object Processing

Like the Signal Processing algorithms, the Object Processing functions employed in EXOSIM Version 2.0 were developed for economy of run time and are confined to a single object or cluster. The algorithms used are: track gating, clustering, area/intensity centroiding, CSO resolve, frame rate select, target designation, imaged (resolved) target logic, and finally, the terminate guidance logic as the target fills the FOV just before impact. Since only one object is being tracked, (within the gated FOV), a

frame-to-frame correlation algorithm is unnecessary. Aimpoint select and edge track algorithms were not developed for this version of EXOSIM, but could be easily included for future enhancements.

5.4.2.6 Simulation Structure

Several changes were made in the simulation structure to allow mapping onto the GIT PFP. These includes:

no constants are passed in argument lists

initialize subroutine on each individual module

option flag to event or time drive seeker initiation

frame rate reciprocal made equal to integral of integration time

5.4.3 Parallel EXOSIM

The simulation described in section 5.4.2 was sent to Dynetics Inc. for conversion to a parallel format. Dynetics completed an analysis of the Boost phase, stages 1 and 2, and sent a five processor version of the code to Georgia Tech. This code became the baseline for all work in developing a parallel implementation of EXOSIM.

Georgia Tech examined all the Dynetics code and partitioned this five processor version into a 27 processor version. This was run on iSBC 286/12 and iSBC 386/12 processors. On the iSBC 386/12 the run time was 3.9:1, which is four times real-time . Efforts were then initiated to convert the code to the GT-FPP/3. This is now underway. All 27 code sections have been converted to C and tested for accuracy. The C code is now being compiled and testing will begin shortly.

Conversion of EXOSIM V2.0 will begin as soon as the work on V1.0 proves all the Software tools are reliable. This version, exploded into all its subroutines, is shown in Figure 5.2. Georgia Tech has separated the subroutines into two flight phases. Figure 5.3 shows the block diagram for the boost phase. This code will follow the version 1.0 code very closely. Modifications should be minimal, resulting in a fast conversion process. The midcourse/terminal phase shown in Figure 5.4 will be an all new effort. Most of this code is quite different from the boost phase and will require a significant effort to convert from serial to parallel form.

5.4.4 Parallel Ada EXOSIM

After a version is running in C on the PFP, it can be rewritten in Ada, translated from Ada to C, compiled and executed. These steps have been done for some small code sections, but not for any large simulation. This work is continuing under USASDC Contract No. DASG60-89-C-0142.

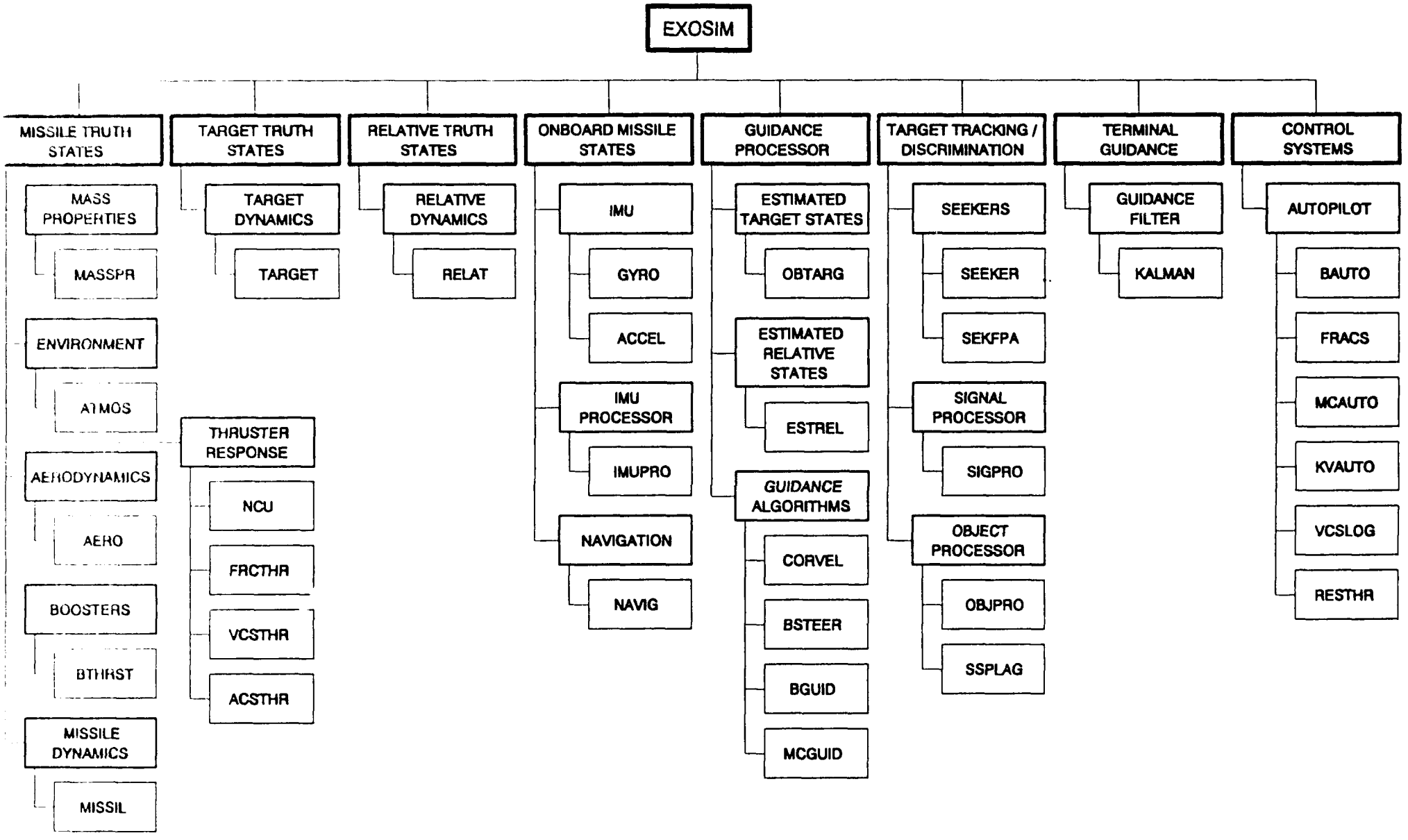


FIGURE 5.2 EXOSIM VERSION 2.0

MG - \WIN30\GORDON\FR1990\EXOSIMV2.DRW

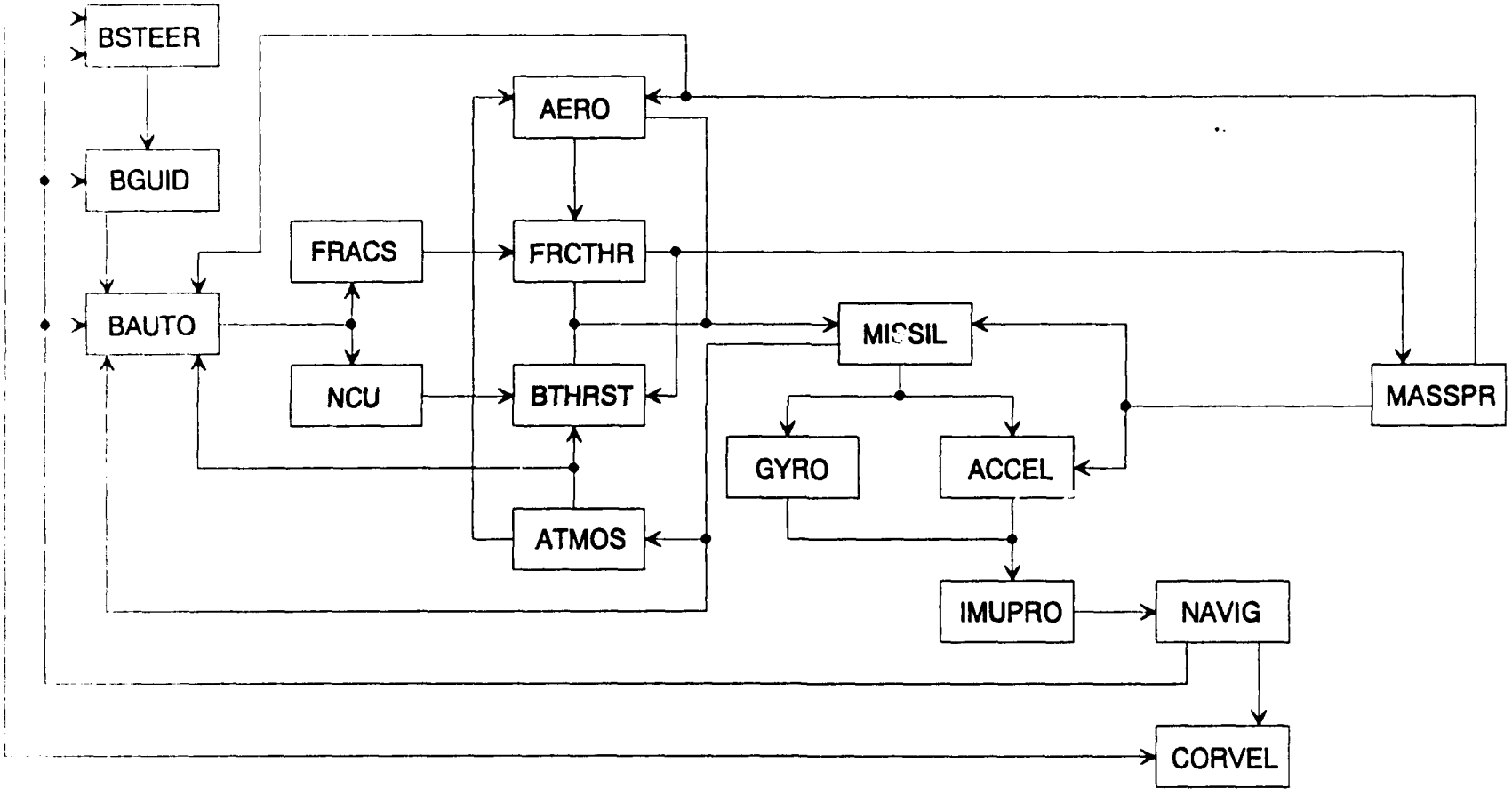


FIGURE 5.3 BOOST PHASE BLOCK DIAGRAM

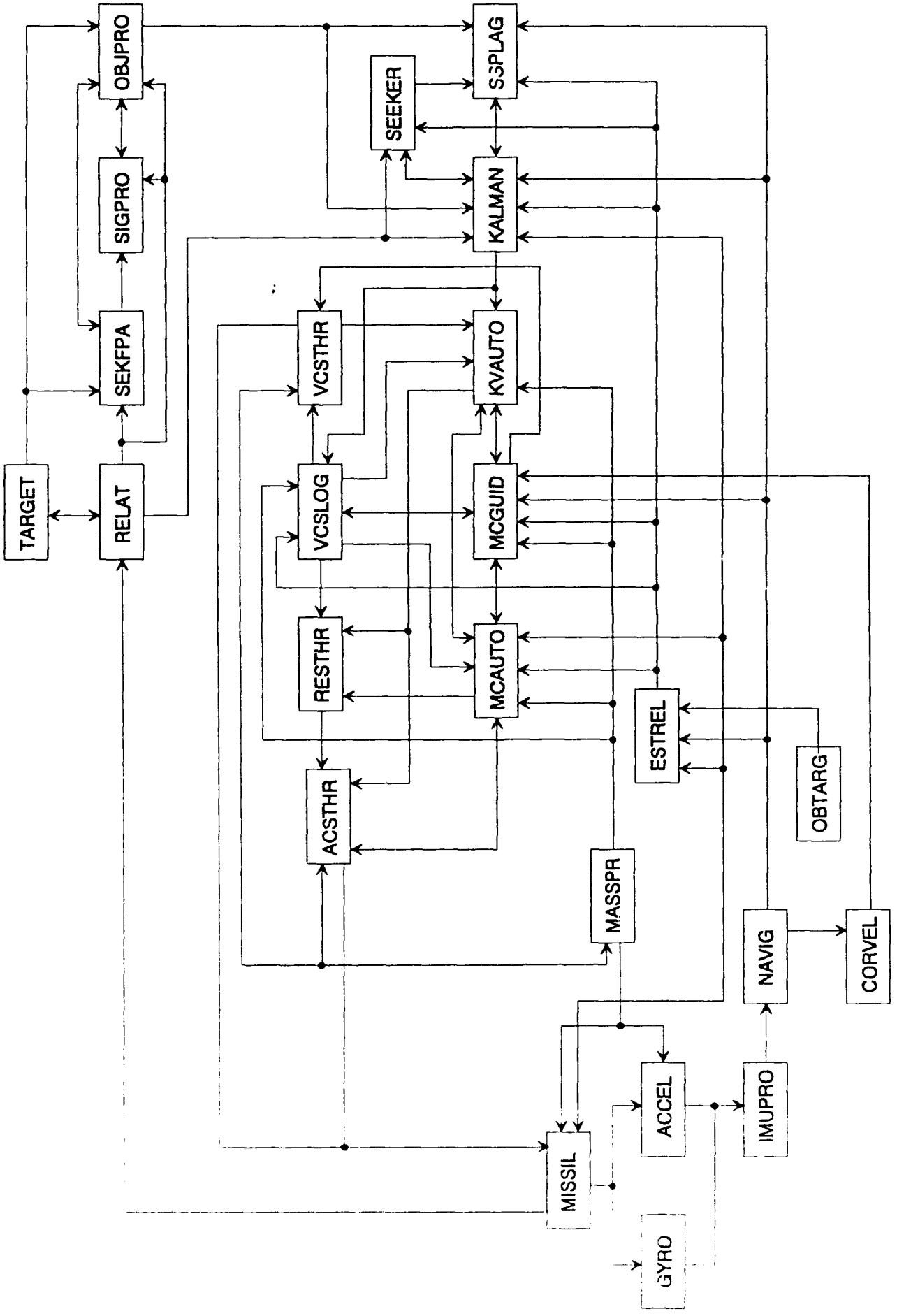


FIGURE 5.4 MIDCOURSE / TERMINAL PHASE BLOCK DIAGRAM

MG - \WIN30\GORDON\FIG1990\MIDCORSE.DRW

6.0 TECHNOLOGY REFERENCES

Technology references are listed in Section 7. This section coordinates the references to a particular technology.

6.1. SEEKER/SCENE EMULATOR TECHNOLOGY

6.1.1. Overview

[16] 9, 44, 49 - 50

6.1.2. Hardware

[16] 45 - 46

6.1.2.1. GT - XIT/1: Crossbar Interface Module Version 1

[10] 30

6.1.2.2. GT - XIT/2: Crossbar Interface Module Version 2

[16] 45, [18] App A

6.1.2.3. GT - SEI/2: Seeker Emulator Interconnect Board

[16] 45, [18] App B

6.1.2.4. Backplane

[16] 45, [18] App C

6.1.3. Software

6.1.3.1. Data Pre-processing

[16] 46, [18] App D

6.1.3.2. SSE Operation

[16] 46, [18] App E,F

6.1.3.3. Data Generation

[18] App G

6.1.3.4. Signal Processing Algorithms

[16] 47 - 49, [18] App H, I, J, K, L, M

6.2. PARALLEL FUNCTION PROCESSOR TECHNOLOGY

6.2.1. Spock

[1] 1.1, 1.5

6.2.1.1. Spock Array

[4] 3.1 - 3.3

6.2.2. Architecture

[9] 29, [16] 11 - 13, [45]

6.2.3. Hardware

6.2.3.1. GT - ADDA/2: Analog Input/Output Module
[9] 30, [16] 16 - 22, [17] App B, [44], [46]

6.2.3.2. GT - ARI/1: Array Interconnect Module
[9] 30, [16] 23 - 25, [17] App C, F, [44], [46]

6.2.3.3. GT - DPX/1: Double Precision Extension for GT-FPP
[20] 80

6.2.3.4. GT-FFS/1: Multi-Function Board for GT-FPP
[20] 80, [26], [44], [46]

6.2.3.5. GT-FPP/1: Floating Point Processor Version 1
[4] 1.1 - 1.11

6.2.3.6. GT-FPP/2: Floating Point Processor Version 2
[9] 31, [20] 79, [34]

6.2.3.7. GT-FPP/3: Floating Point Processor Version 3
[44], [46]

6.2.3.8. GT-FXM/1: Multi Variable Function Processor
[9]48 - 53, [11] App A,B

6.2.3.9. GT-FXS/1: Single Variable Function Processor
[1]2.1 - 2.21, [9]31 - 41

6.2.3.10. GT-SEQ/2: Sequencer Module
[9] 42 - 48, [11] App N, [16] 29 - 37, [44], [46]

6.2.3.11. GT-SPT/1: Sequencer/Processor Transition Board Version 1
[44], [46]

6.2.3.12. GT-SPT/2: Sequencer/Processor Transition Board Version 2
[9] 48

6.2.3.13. SUN Host
[16] 43

6.2.3.14. GT - SSD/1:
[9] 48

6.2.3.15. GT-SXI/2: Sequencer/Crossbar Interface Version 2
[9] 48, [44], [46]

6.2.3.16. GT-SXI/3: Sequencer/Crossbar Interface Version 3
[16] 25 - 26, [17] App D

6.2.3.17. GT-XB2/1: Crossbar Baord
[16] 13 - 15, [17] App A, [44], [46]

6.2.3.18. GT-DXB/2: Crossbar Piggyback Board
[44], [46]

6.2.3.19. GT-XI 86/2:
[9] 30

6.2.3.20. GT-XI 286/1: Crossbar Interface/iSBC 286/Version 1
[44], [46]

6.2.3.21. GT-XI 286/2: Crossbar Interface/iSBC 286/Version 2
[16] 26 - 28, [17] App E,F

6.2.3.22. GT-XIT/1:
[9] 31, [10] 30

6.2.3.23. GT-XSD/2: Crossbar Status Display Board
[44], [46]

6.2.3.24. GT-MRH/1: Multibus Repeater Host
[44], [46]

6.2.3.25. GT-MRS/1: Multibus Repeater Slave
[44], [46]

6.2.3.26. GT-DT2/1: Data Memory Board/GT-FPP
[44], [46]

6.2.4. Software

6.2.4.1. Test Routines
[1] 1.1 - 1.4, [16] 51 - 53, [19] App B, [47]

6.2.4.2. Chromatics Display
[1] 1.5

6.2.4.3. Host Utilities
[1] A.1 - A.48

6.2.4.4. Crossbar Interface Utilities for iSBC 286/12
[9] 55, [11] App H

6.2.4.5. Ada Utility Routine
[20] App A

6.2.4.6. System Software
[16] 54, [19] App C

6.2.4.7. Pascal Compiler for GT - FPP
[9] 54, [35]

6.2.4.8. Crossbar Compiler
[9] 55, [11] App G

6.2.4.9. Simulator for GT-FPP
[4] App B

6.2.4.10. Microassembler for GT-FPP

[4] App A

6.2.4.11. Support Utilities for GT-FXS/1
[9] 54, [11] App F

6.2.4.12. PFP Simulation Support
[9] 54

6.2.4.13. PFP Programming Environment
[31]

6.2.4.14. PFP Run Time Kernel
[25]

6.2.4.15. iSBC 286/12 Monitor
[16]51, [19] App A, [44]

6.2.5. Automated Input

6.2.5.1. P-CAD
[1] 1.7 - 1.10

6.2.5.2. Block Diagram Input
[9] 59 - 76, [11] App C, D

6.3. DIGITAL EIMULATION TECHNOLOGY LABORATORY

6.3.1. Overview
[16] 1-7, 9

6.3.2. Simulation

6.3.2.1. DOF Linear EXO Interceptor model (Dynetics)
[2] 4.10 - 4.36, [9] 56, [11] App. I

6.3.2.2. DOF Linear ENDO Interceptor Model (Dynetics)
[2] 4.36 - 4.42, [4] 3.4 - 3.33

6.3.2.3. DOF Nonlinear ENDO Interceptor Model (Dynetics)
[2] 4.42 - 4.71, [4] 3.34 - 3.10

6.3.2.4. Satellite Attitude Control System
[9] 56, [11] App J

6.3.2.5. Target Initialization
[16] 56, [19] App D

6.3.2.6. Spinning Missile
[16] 56, [19] App E

6.3.2.7. EXOSIM
[16] 15, [40], [41], [42], [43]

6.3.3. Benchmarks

6.3.3.1. Fast Fourier Transform
[9] 56 - 57, [11] App J

6.3.3.2. Matrix Multiplication
[9] 67 - 58, [11] App L

6.3.3.3. Partial Differential Equations
[9] 58, [11] App O

6.4. GUIDANCE, NAVIGATION AND CONTROL PROCESSOR TECHNOLOGY

6.4.1. Overview
[20] 1, [38] 1 - 2, [15] 1 - 2

6.4.2. GT-DP/1: Data Processor
[5] 3 - 4, [12] 6 - 7, [13] 5 - 6, [15] 1 - 2

6.4.2.1. Architecture
[10] 31 - 33, 35 - 36, [20] 2

6.4.2.2. GT-VDR/1: Dataram
[4] 2.13 - 2.15, [10] 42 - 46, [20] 4, [27], [28], [5] 35 - 48, [13] 10 - 13

6.4.2.3. GT-VFPU/1: Floating Poing Unit Version 1
[4] 2.13, [10] 47 - 48

6.4.2.4. GT-VFPU/2: Floating Point Unit Version 2
[20] 4, [29], [30], [13] 14 - 18

6.4.2.5. GTVHI/1: Host Interface
[4] 2.3, [9] 93 - 94

6.4.2.6. GT-VSEQ/1: Sequencer
[4] 2.3 - 2.7 [10] 41 - 43, [20] 2, [23], [24], [5] 6 - 34, [13] 7 - 9

6.4.2.7. GT-VSM8/1: Serial Network Interface
[9] 90 - 92, [10] 49 - 52, [20] 4, [32], [33], [15] 9, [13] 24 - 28, [5] 61, [12] 22 - 26, [13] 19 - 23

6.4.2.8. GT-VSNI/1: Serial Switch Matrix
[10] 37 - 40, [20] 4, [21], [22], [5] 49 - 62, [12] 27 - 28

6.4.2.9. Multichip Simulation
[13] 29 - 30

6.4.3. GT-SP: Signal Processor

6.4.3.1. GT-VCLS/1: Clustering
[20] 67 - 70, [15] 5 - 6

6.4.3.2. GT-VCTR/1: Centroiding
[20] 71 - 76, [15] 6 - 7

6.4.3.3. GT-VFPI/1: FPA Interface

[20] 5 - 6

6.4.3.4. 6.4.3.4.GT - VGS/1: Gamma Supperssion
[20] 5

6.4.3 5. GT-VNUC/1: Non-uniformity Correction
[20] 8 - 21, [15] 2 - 3

6.4.3.6. GT-VTF/1: Temporal Filtering
[20] 22 - 45, [15] 3

6.4.3.7. GT-VSF/1: Spatial Filter
[20] 46 - 52, [15] 4

6.4.3.8. GT-VTHR/1: Thresholding
[20] 53 - 66, [15] 4 - 5

6.4.4. GT-EP: Executive Processor
[15] 7 - 8

6.4.4.1. GT-VDAG/1: Data Address Generator
[20] 78, [37], [15] 8 - 9

6.4.4.2. GT-VIAG/1: Instruction Address Generator
[20] 77, [36], [15] 8

6.4.5. VLSI Design Testing
[10] 53 - 54, [12] 1 - 4

6.4.6. Transputer GN&C Processor

6.4.6.1. GT-TP8/1: Transputer Prototype Version 1
[1] 2.28 - 2.36, App B, [9] 48, [10] 26 - 27

6.4.6.2. GT-TP8/2: Transputer Prototype Version 2
[19] 27 - 30

6.4.6.3. Transputer Interconnect Chip
[5] 6 - 10

6.5. VLSI PARALLEL FUNCTION PROCESSOR TECHNOLOGY
[5] 1 - 2, [12] 6 - 7

6.5.1. Architecture
[9] 78 - 79

6.5.2. VLSI Design

6.5.2.1. GT-VHC/1: Crossbar Handshake Controller
[4] 2.16 - 2.20, [9] 64 - 86, [5] 20 - 49, [12] 13 - 16

6.5.2.2. GT-VSM32/1: Switch Matrix
[4] 2.20 - 2.26, [9] 80 - 83, [5] 11 - 19, [12] 17 - 21

6.5.2.3. GT-VPNI/1: Parallel Network Interface
[4] 2.7 - 2.13, [9] 87 - 89, [5] 50 - 60, [12] 8 - 12

6.5.2.4. GT-FCS/1: Fully Connected Switch
[15] 10

6.5.2.5. GT-NTC/1: Network Traffic Controller
[15] 10 - 11

6.6. MISCELLANEOUS

6.6.1. Specifications

6.6.1 1. Crossbar Interface Standard
[11] App E

6.6.2. FPA Technology

6.6.2.1. IR Seeker Characteristics
[2] 7.24 - 7.52

6.6.2.2. Seeker Design
[2] 7.53 - 7.69

6.6.3. Component Technology
[2] 7.94 - 7.101

6.6.4. Guidance
[2] 7.2 - 7.23

6.6.5. Trajectory Modeling and Simulation
[2] 4.73 - 4.120, [2] 7.70 - 7.93

6.6.6. Materials Management
[16] 37 - 42

7.0 REFERENCES

[1] Alford, C.O. and J.O. Hamblen, "Macrostructure Logic Arrays," Volume 1, Final Technical Report, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, June 20, 1986.

[2] Alford, C.O. and J.O. Hamblen, "Macrostructure Logic Arrays," Volume 2, Final Technical Report, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, June 20, 1986.

[3] Alford, C.O. and J.O. Hamblen, "Macrostructure Logic Arrays," Volume 3, Final Technical Report, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, June 20, 1986.

[4] Alford, C.O. and J.O. Hamblen, "Macrostructure Logic Arrays," Volume 4, Final Technical Report, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, June 20, 1986.

[5] Alford, C.O. and J.O. Hamblen, "VLSI Development Plan," Volume 1, Interim Technical Report, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, July 15, 1986.

[6] Alford, C.O. and J.O. Hamblen, "VLSI Development Plan," Volume 2, Interim Technical Report, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, July 15, 1986.

[7] Alford, C.O. and J.O. Hamblen, "Program Plan," Volume 1, Interim Technical Report, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, July 15, 1986.

[8] Alford, C.O. and J.O. Hamblen, "Program Plan," Volume 2, Interim Technical Report, U.S. Army Strategic Defense Command, Contract No. Dasg60-85-C-0041, Georgia Tech, CERL, July 15, 1986.

[9] Alford, C.O., "Macrostructure Logic Arrays," Volume 1, Final Technical Report, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, November 10, 1987.

[10] Alford, C.O., "Macrostructure Logic Arrays," Volume 2, Final Technical Report, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, November 10, 1987.

[11] Alford, C.O., "Macrostructure Logic Arrays," Volume 3, Final Technical Report, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, November 10, 1987.

[12] Alford, C.O., "VLSI Development Plan," Volume 1, Interim Technical Report, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, November 25, 1987.

[13] Alford, C.O., "VLSI Development Plan," Volume 2, Interim Technical Report, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, November 25, 1987.

[14] Alford, C.O., "Program Plan - CLIN 0005," Interim Technical Report, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, July 1, 1989.

[15] Alford, C.O., "VLSI Development Plan - CLIN 0005," Interim Technical Report, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, June 9, 1989.

[16] Collins, T.R., et. at., "Macrostructure Logic Arrays," Volume 1, Final Technical Report, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, July 20, 1989.

[17] Collins, T.R., et. al., "KEW Digital Emulation Laboratory," Volume 2, Final Technical Report, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, July 20, 1989.

[18] Collins, T.R., et. al., "KEW Digital Emulation Laboratory," Volume 3, Final Technical Report, U.S. Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, July 20, 1989.

[19] Collins, T.R., et. al., "KEW Digital Emulation Laboratory," Volume 4, Final Technical Report, U.S. Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, July 20, 1989.

[20] Collins, T.R., et. al., "GN&C Processor Development," Volume 5, Final Technical Report, U.S. Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, July 20, 1989.

[21] Ghori, A., "Eight Point Crossbar Switch Chip GT-VSM8/1:VLSI Design Document," CERL Technical Report, CERL008-0020.1, Georgia Tech, CERL, May 26, 1988.

[22] Ghori, A., "Eight Point Crossbar Switch Chip Gt-VSM8/1:Design Verification Document," Special Technical Report CERL008-0021.1, U.S. Army Strategic Defense Command, Contract No. DASG60-845-C-0041, Georgia Tech, CERL, May 26, 1988.

[23] Ghori, A., "Sequencer GT-VSEQ/1: VLSI Design Document," CERL Technical Report, CERL008-0040.1, Georgia Tech, CERL, March 26, 1989.

[24] Ghori, A., "Sequencer GT-VSEQ/1: VLSI Design Verification Document," Special Technical Report, CERL008-0041.1, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, March 26, 1989.

[25] Schwan, Karsten et. al., "Parallel Function Processor Run-time Kernel (DRAFT): Software Design Document," Special Technical Report CERL003-0030.1, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, June 28, 1989.

[26] Register, A., "High Speed Function Board GT-FFS/1: Hardware Design Document" Special Technical Report CERL001-0002.1, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, August 5, 1988.

[27] Russ, S., "Instruction and Data Memory Unit GT-VDR/1: VLSI Design Document," CERL Technical Report, CERL008-0050.1, Georgia Tech, CERL, August 17, 1988.

[28] Russ, S., "Instruction and Data Memory unit GT-VDR/1: Design Verification Document," Special Technical Report CERL008-0051.1, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, August 17, 1988.

[29] Russ, S., "Floating/Fixed Point ALU Unit GT-VFPA/1: VLSI Design Document," CERL Technical Report, CERL008-0030.1, Georgia Tech, CERL, August 17, 1988.

[30] Russ, S., "Floating/Fixed Point ALU Unit GT-VFPA/1: VLSI Design Verfication Document," Special Technical Report CERL008-0031.1, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL August 17, 1988.

[31] Shilling, John et. at., "Parallel Function Processor Programming Environment: Software Design Document," Special technical Report 003-0020.1, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, June 28, 1989.

[32] Tan, W.S., "Serial Network Interface GT-VSNI/1: Design Document," CERL Technical Report CERL008-0010.1, Georgia Tech, CERL, May 12, 1988.

[33] Tan, W.S., "Serial Network Interface GT-VSNI/1: Design Verification Document," CERL Technical Report, CERL008-0011.1, Georgia Tech, CERL, May 12, 1988.

[34] Tan, W.S., "G&C High Speed Floating Point Processor Design," Special Technical Report CERL001-0001.1, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, June 15, 1988.

[35] Tan, W.S., "G&C High Speed Floating Point Processor Compiler Design," Special Technical Report CERL003-0001.1, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, June 15, 1988.

[36] Tan, W.S., "Instruction Address Generation GT-VIAG/1: Programming Model Document," Special Technical Report CERL008-0062.1, U.S. Army Strategic

Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, November 5, 1988.

[37] Tan, W.S., "Data Address Generation GT-VDAG: Programming Model Document," Special Technical Report CERL008-00621.1, U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, November 5, 1988.

[38] Tan, W.S., "VLSI Development Plan," Special Technical Report, CERL008-0140.1. U.S. Army Strategic Defense Command, Contract No. DASG60-85-C-0041, Georgia Tech, CERL, June 9, 1989.

[39] Levinson, L.M. et. al., "High Density Interconnects Using Laser Lithography," *Proceedings of the National Electronics Packaging and Production Conference*, March 6 - 9, 1989, Anaheim, CA, pp 1319 - 1328.

[40] _____, "EXOSIM Version 1.0 Simulation Description, Volume 1," Interim Technical Report, U.S. Army Strategic Defense Command, Contract No. DASG60-88-C-0002, Coleman Research Corporation, 30 June 1989.

[41] _____, "EXOSIM Version 1.0 Simulation Description, Volume 2," Interim Technical Report, U.S. Army Strategic Defense Command, Contract No. DASG60-88-C-0002, Coleman Research Corporation, 30 June 1989.

[42] _____, "EXOSIM Version 2.0 Simulation Description, Volume 1," Interim Technical Report, CHR/89-2176, U.S. Army Strategic Defense Command, Contract No. DASG60-88-C-0002, Coleman Research Corporation, 30 October 1989.

[43] _____, "EXOSIM Version 2.0 Simulation Description, Volume 2," Interim Technical Report, CHR/89-2176, U.S. Army Strategic Defense Command, Contract No. DASG60-88-C-0002, Coleman Research Corporation, 30 October 1989.

[44] _____, "Parallel Function Processor Technical Data package," Vol.1, Interim Technical Report, U.S. Army Strategic Defense Command, Contract No. DASG60-89-C-0142, Georgia Tech, CERL, March 31, 1990.

[45] _____, "Parallel Function Processor Technical Data package," Vol.2, Interim Technical Report, U.S. Army Strategic Defense Command, Contract No. DASG60-89-C-0142, Georgia Tech, CERL, March 31, 1990.

[46] _____, "Parallel Function Processor Technical Data package," Vol.3, Interim Technical Report, U.S. Army Strategic Defense Command, Contract No. DASG60-89-C-0142, Georgia Tech, CERL, March 31, 1990.

[47] _____, "Parallel Function Processor Technical Data package," Vol.4, Interim Technical Report, U.S. Army Strategic Defense Command, Contract No. DASG60-89-C-0142, Georgia Tech, CERL, March 31, 1990.